SUPPLEMENT TO

Beneath Apple ProDOS

For ProDOS Versions 1.0.1 and 1.0.2

by Don D. Worth and Pieter M. Lechnner

QUALITY SOFTWARE
21601 Marilla Street
Chatsworth, California 91311
Apple Books from Quality Software

Beneath Apple DOS
by Don Worth & Pieter Lechner

$19.95

Understanding the Apple II
by Jim Sather

$22.95

Understanding the Apple IIe (Available Nov. 1984)
by Jim Sather

$24.95

Apple Utility Software from Quality Software

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

$39.95

Universal File Conversion (includes diskette)
by Gary Charpentier

$34.95

Production Editor: Kathryn M. Schmidt
Illustrations by: George Garcia

(c) 1984 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>TOPIC</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>5</td>
</tr>
<tr>
<td>Understanding the Listings</td>
<td>5</td>
</tr>
<tr>
<td>Disk Controller Boot ROM--Apple II/II+/IIe</td>
<td>6</td>
</tr>
<tr>
<td>Disk Controller Boot ROM--Apple IIC</td>
<td>8</td>
</tr>
<tr>
<td><strong>ProDOS VERSION 1.0.1:</strong></td>
<td></td>
</tr>
<tr>
<td>ProDOS Loader</td>
<td>11</td>
</tr>
<tr>
<td>ProDOS Relocator</td>
<td>14</td>
</tr>
<tr>
<td>(includes /RAM device driver and BI loader)</td>
<td></td>
</tr>
<tr>
<td>ProDOS MLI (Kernel)</td>
<td>27</td>
</tr>
<tr>
<td>ProDOS System Global Page</td>
<td>61</td>
</tr>
<tr>
<td>ProDOS Quit Code</td>
<td>63</td>
</tr>
<tr>
<td>ProDOS Disk II Device Driver</td>
<td>67</td>
</tr>
<tr>
<td>ProDOS IRQ Handler</td>
<td>74</td>
</tr>
<tr>
<td>ProDOS BI Relocator</td>
<td>75</td>
</tr>
<tr>
<td>ProDOS BASIC Interpreter (BI)</td>
<td>78</td>
</tr>
<tr>
<td>ProDOS BI Global Page</td>
<td>114</td>
</tr>
<tr>
<td><strong>ProDOS VERSION 1.0.2</strong></td>
<td></td>
</tr>
<tr>
<td><strong>APPENDIXES</strong></td>
<td></td>
</tr>
<tr>
<td>Errata to <em>Beneath Apple ProDOS</em></td>
<td>121</td>
</tr>
<tr>
<td>1st printing, 1984</td>
<td></td>
</tr>
<tr>
<td>Ordering Future Supplements</td>
<td>124</td>
</tr>
</tbody>
</table>
A Pro DOS ATLAS
INTRODUCTION

This supplement documents the actual structure and logic of the ProDOS system at nearly a byte by byte level. It is intended to aid experienced programmers in designing customized interfaces to ProDOS, and to provide implicit documentation of ProDOS's functions. Less advanced assembly language programmers may find this supplement useful in learning about how an operating system works. Providing this information does not constitute an endorsement by the authors of indiscriminant modification of the ProDOS components. Whenever possible, standardized interfaces to ProDOS should be used to avoid the uncontrolled modifications which were made to DOS 3.3.

External system programs and utilities such as the FILER and CONVERT are not covered here.

The information provided here is for two releases of the ProDOS operating system--Version 1.0.1 and Version 1.0.2. Because these versions are so similar, we have included both in the same supplement. Version 1.0.1 is first presented in its complete form. Then Version 1.0.2 is presented by pointing out and documenting those areas that are different from Version 1.0.1.

As new releases of ProDOS become available, additional supplements to Beneath Apple ProDOS will be prepared. To order supplements for other versions of ProDOS, fill out the order form on page 125 of this supplement. Ordering instructions can be found on page 124. When ordering a new supplement, be sure to specify the version of ProDOS you want the new supplement for.

UNDERSTANDING THE LISTINGS

The listings which follow describe the major ProDOS 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.
Beneath Apple ProDOS Supplement

Disk Controller Boot ROM -- Apple II/Ile II+ IIe NEXT OBJECT ADDR: C600

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>C600</td>
<td>MODULE STARTING ADDRESS</td>
</tr>
<tr>
<td></td>
<td>***********************</td>
</tr>
<tr>
<td></td>
<td>* BOOT ROM -- APPLE DISK CONTROLLER *</td>
</tr>
<tr>
<td></td>
<td>* THIS CODE RESIDES FROM $C600</td>
</tr>
<tr>
<td></td>
<td>TO $C6FF, IT LOADS TRACK 0</td>
</tr>
<tr>
<td></td>
<td>* SECTOR 0 INTO RAM AT $8000 AND</td>
</tr>
<tr>
<td></td>
<td>* JUMPS TO IT</td>
</tr>
<tr>
<td></td>
<td>* VERSION 1.0.1 -- 1 JAN 84</td>
</tr>
<tr>
<td></td>
<td>***********************</td>
</tr>
<tr>
<td></td>
<td>********** ZERO PAGE ADDRESSES **********</td>
</tr>
<tr>
<td>0026</td>
<td>SECTOR BUFFER POINT</td>
</tr>
<tr>
<td>0028</td>
<td>SLOT NUMBER * 16 FOR INDEX</td>
</tr>
<tr>
<td>003C</td>
<td>WORKBYTE</td>
</tr>
<tr>
<td>003E</td>
<td>SECTOR WANTED</td>
</tr>
<tr>
<td>0040</td>
<td>TRACK FOUND</td>
</tr>
<tr>
<td>0041</td>
<td>TRACK WANTED</td>
</tr>
<tr>
<td>0190</td>
<td>SYSTEM STACK</td>
</tr>
<tr>
<td>0200</td>
<td>TRANSLATE TABLE - $30</td>
</tr>
<tr>
<td>0300</td>
<td>AUXILIARY BUFFER</td>
</tr>
<tr>
<td>0356</td>
<td>TRANSLATE TABLE</td>
</tr>
<tr>
<td>0800</td>
<td>SECTORS TO LOAD</td>
</tr>
<tr>
<td>0801</td>
<td>ENTRY POINT</td>
</tr>
<tr>
<td>0880</td>
<td>PHASES OFF</td>
</tr>
<tr>
<td>0881</td>
<td>PHASES ON</td>
</tr>
<tr>
<td>0889</td>
<td>MOTOR ON</td>
</tr>
<tr>
<td>C08A</td>
<td>DRIVE SELECT</td>
</tr>
<tr>
<td>C08C</td>
<td>READ DATA REGISTER</td>
</tr>
<tr>
<td>C08E</td>
<td>SET READ MODE</td>
</tr>
<tr>
<td>FC0A</td>
<td>MONITOR WAIT ROUTINE</td>
</tr>
<tr>
<td>FF58</td>
<td>RTS</td>
</tr>
</tbody>
</table>

C600 ********** BUILD READ TRANSLATE TABLE **********

C600 SIGNATURE
C602 INITIALIZE TABLE VALUE INDICATOR
C606 STORE BIT PATTERN
C609 SHIFTPATTERNLEFTONEBIT
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

Disk Controller Boot ROM -- Apple II/Ile II+ IIe NEXT OBJECT ADDR: C614

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>C614</td>
<td>--- &gt;&gt;C61E</td>
</tr>
<tr>
<td>C616</td>
<td>SHIFT PATTERN RIGHT, MUST HAVE ONLY ONE BIT ON</td>
</tr>
<tr>
<td>C617</td>
<td>IF MORE THAN ONE BIT ON, TRY ANOTHER PATTERN &gt;&gt;C614</td>
</tr>
<tr>
<td>C619</td>
<td>FOUND ONE, GET TABLE VALUE</td>
</tr>
<tr>
<td>C61A</td>
<td>AND STORE IT IN TABLE (8356)</td>
</tr>
<tr>
<td>C61D</td>
<td>INCREMENT TABLE VALUE INDICATOR</td>
</tr>
<tr>
<td>C61E</td>
<td>GET NEXT BIT PATTERN, DONE YET</td>
</tr>
<tr>
<td>C61F</td>
<td>NO, GO CHECK IT OUT &gt;&gt;C686</td>
</tr>
</tbody>
</table>

C621 ********** DETERMINE SLOT, TURN DRIVE ON **********

C621 CALL A KNOWN RTS <FF50>
C624 CALL STACK POINT |
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 |
C62F INSURE READ MODE (C08E) |
C635 SELECT DRIVE 1 (C08A) |
C638 TURN THE MOTOR ON (C089) |

C63B ********** RECALIBRATE DISK ARM **********

C63B PREPAIR TO STEP THE ARM 80 PHASES |
C63D TURN A PHASE OFF (C080) |
C640 PUT COUNTER IN ACCUMULATOR |
C641 CREATE A PHASE NUMBER (8-3) |
C643 DOUBLE IT FOR PROPER INDEX |
C644 COMBINE WITH SLOT FOR FINAL INDEX |
C646 PUT INDEX IN X REGISTER |
C647 TURN A PHASE ON (C081) |
C648 DELAY ABOUT 20 MICROSCONDS |
C64F DECREMENT COUNTER |
C650 LOOP UNTIL ALL 80 ARE DONE >>C63D |

C652 ********** INITIALIZATION **********

C652 --- |
C654 SECTOR TO FIND -> $00 |
C656 TRAFFIC TO FIND -> $00 |
C65A MAIN BUFFER POINTER ($26) -> $8880 |
C65C CLEAR THE CARRY |
C65D PUSH STATUS ON STACK |

C65E ********** SEARCH FOR A VALID HEADER **********

C65E CHECK DATA REGISTER (C08C) |
C661 LOOP UNTIL DATA IS VALID >>C65E |
C663 IS IT 00? |
C665 NO, TRY AGAIN >>C65E |
C667 YES, CHECK REGISTER AGAIN (C08C) |
<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>C66A</td>
<td>LOOP UNTIL VALID &gt;&gt;C667</td>
</tr>
<tr>
<td>C66C</td>
<td>IS IT AN SAA</td>
</tr>
<tr>
<td>C66D</td>
<td>NO, SEE IF ITS A $DD &gt;&gt;C663</td>
</tr>
<tr>
<td>C66F</td>
<td>YES, DELAY FOR REGISTER TO CLEAR</td>
</tr>
<tr>
<td>C671</td>
<td>CHECK REGISTER (C08C)</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 SAD?</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;C6C5</td>
</tr>
</tbody>
</table>

### Disk Controller Boot ROM -- Apple II/II+//IIE NEXT OBJECT ADDR: C6A6

<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 (C08C)</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 (C08C)</td>
</tr>
<tr>
<td>C692</td>
<td>LOOP UNTIL VALID &gt;&gt;C69F</td>
</tr>
<tr>
<td>C694</td>
<td>COMBINE WITH PREVIOUS X1X1X1X1 AND X1X1X1X1</td>
</tr>
<tr>
<td>C696</td>
<td>DECREMENT COUNTER, DONE YET?</td>
</tr>
<tr>
<td>C697</td>
<td>NO, DO ANOTHER &gt;&gt;C6A5</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>

### Disk Controller Boot ROM -- Apple II/II+//IIE NEXT OBJECT ADDR: C6C9

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>C6C9</td>
<td>LOOP UNTIL BUFFER FULL &gt;&gt;C6BA</td>
</tr>
<tr>
<td>C6CB</td>
<td>READ DATA REGISTER (C08C)</td>
</tr>
<tr>
<td>C6CE</td>
<td>LOOP UNTIL VALID &gt;&gt;C6CB</td>
</tr>
<tr>
<td>C6D0</td>
<td>IS CHECKSUM OKAY? ($2D6)</td>
</tr>
<tr>
<td>C6D3</td>
<td>NO, START OVER &gt;&gt;C65C</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>DECURRENT OFFSET (AUXILIARY 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>C6E5</td>
<td>SAVE COMPLETED DATA BYTE</td>
</tr>
<tr>
<td>C6E8</td>
<td>INCURRENT BUFFER (MAIN BUFFER)</td>
</tr>
<tr>
<td>C6EA</td>
<td>LOOP UNTIL WHOLE BUFFER IS DONE &gt;&gt;C609</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? ($900)</td>
</tr>
<tr>
<td>C6F5</td>
<td>YES, GO DO IT &gt;&gt;C6D3</td>
</tr>
<tr>
<td>C6F8</td>
<td>NO, ENTER CODE WE JUST LOADED &gt;&gt;$901</td>
</tr>
</tbody>
</table>

### C6FB

************ UNUSED ************

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>C6FB</td>
<td>---</td>
</tr>
</tbody>
</table>
Beneath Apple ProDOS Supplement

Disk Controller Boot ROM -- Apple IIc

NEXT OBJECT ADDR: C600

ADDR DESCRIPTION/CONTENTS

C600 MODULE STARTING ADDRESS

******************************************************************************
* BOOT ROM - APPLE //C CONTROLLER ROM *
* THIS CODE RESIDES FROM SC600 *
* TO SC73F, IT LOADS TRACK 0 *
* SECTOR 0 INTO RAM AT $800 AND *
* JUMPS TO IT. IT CAN BOOT FROM *
* DRIVE 2 AND HAS SOME MINIMAL *
* ERROR CHECKING *
* *
* VERSION 1.0.1 -- 1 JAN 84 *
* *
******************************************************************************

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

@@@3 RETRY COUNT (HIGH BYTE)
@@26 SECTOR BUFFER POINTER
@@2B SLOT NUMBER * 16 FOR INDEX
@@3C WORKBYTE
@@3D SECTOR WANTED
@@40 TRACK FOUND
@@41 TRACK WANTED
@@4F DRIVE TO BOOT FROM

********** EXTERNAL ADDRESSES **********

$2D6 TRANSLATE TABLE - $80
$300 AUXILIARY BUFFER
$356 TRANSLATE TABLE
$7DB SCREEN LOCATION
$800 SECTORS TO LOAD
$801 ENTRY POINT
C600 PHASE# OFF
C601 PHASE# ON
C608 MOTOR OFF
C609 MOTOR ON
C60C READ DATA REGISTER
C60E SET READ MODE
C6EA DRIVE SELECT
FCA8 MONITOR WAIT ROUTINE

C600 ********** INITIALIZATION **********

C600 SIGNATURE
C602 SET DRIVE -> 1
C604 INITIALIZE RETRY COUNT (HIGH BYTE)

Disk Controller Boot ROM -- Apple IIc

NEXT OBJECT ADDR: C606

ADDR DESCRIPTION/CONTENTS

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 (C08E)
C616 GET DRIVE NUMBER BACK
C617 SELECT APPROPRIATE DRIVE (C0EA)
C61A TURN MOTOR ON (C089)

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 (0-3)
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 -> $00
C638 TRACK TO FIND -> $00
C63A BUILD THE TRANSLATE TABLE <C709>

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
C644 DECREMENT RETRY COUNT, TRY AGAIN?
C646 YES, GO DO IT >>C656
C648 NO, TURN DRIVE OFF (C088)
C64B GET A CHARACTER FROM ERROR MESSAGE (C6CF)
C64E HANG WHEN DONE PRINTING >>C64B
C650 PUT A CHARACTER ON THE SCREEN ($77B)
C653 INCREMENT OFFSET INTO MESSAGE
C654 GO BACK FOR MORE >>C64B
C656 ---
C657 DECREMENT RETRY COUNT (LOW BYTE)
C658 IF NOT ZERO, TRY AGAIN >>C65E
Boot ROM -- Apple IIc

Option/Contents

Beneath Apple

GO DECREMENT RETRY COUNT (HIGH B) FILLER TO POSITION CODE BELOW >>

*** SEARCH FOR A VALID HEADER ***

Disk Control

DATA REGISTER (C88C)
ADDR DESC UNTIL DATA IS VALID >>C65E

TRY AGAIN >>C657

C65A IF S CHECK REGISTER AGAIN (C88C)

C65C SPAC UNTIL VALID >>C667

C661 = NO, IS IT A $86

C665 IF S WE FOUND AN ADDRESS HEADER >>C683

C667 YES, HAVE WE FOUND ONE PREVIOUSLY?

C66A LOOP, START OVER >>C63F

C66C IS IT AN SAD?

C66E NO, WE FOUND A DATA HEADER >>C6A6

C670 YES, START OVER >>C63F

C671 CHECK *** DECODE ADDRESS FIELD ***

C674 LOOP *** DECODE ADDRESS FIELD ***

C676 IS IT $00?

C678 YES, ALIZE COUNTER

C67A NO, VALUE DECODED, WILL BE TRACK ON LANCE

C67B IF S DATA REGISTER (C88C)

C67D WAS UNTIL DATA VALID >>C687

C67F YES, BITS INTO POSITION XIXXIX

C681 NO, FOR LATER

C683 UNTIL VALID >>C68F

C685 SET WITH PREVIOUS XIXXIX AND XIX1 AHEAD FIELD >>C642

C683 INCREMENT COUNTER, DONE YET?

C685 SAVE NO ANOTHER >>C685

C687 READ THE STACK CLEAN

C688 CSH START OVER >>C63F

C68D START TRACK FOUND

C68F READY TRACK WANT?

C692 LOOP, START OVER >>C63F

C694 COML INDICATE ADDRESS FOUND, GO LOOK FC63:

C696 DECE

C697 NO, *** READ DATA FIELD ***

C699 KEE

C69A IS

C69C NO,

C69E GET

C6A0 IS

C6A2 NO,

C6A4 YES

C6A6

C6A8 READ DATA REGISTER (C88C)

C6AD LOOP UNTIL VALID >>C6AA

C6AF EXCLUSIVE-OR WITH TRANSFORM TABLE ($26)

C6BF DECREMENT OFFSET

C6B5 STORE BYTE IN AUXILIARY BUFFER ($00)

C6B8 LOOP UNTIL BUFFER FULL >>C6BA

C6B9 STORE BYTE IN MAIN BUFFER

C6C0 INCREMENT OFFSET

C6C8 STORE BUFFER FULL >>C6C9

C6CA READ DATA REGISTER (C88C)

C6CC LOOP UNTIL VALID >>C6CB

C6D0 IS CHECKSUM OKAY? ($26)

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 >>C677

C6D7 GET BYTE FROM MAIN BUFFER

C6ED ROLL IN TWO BITS FROM AUXILIARY BUFFER

C6EE SAVE COMPLETED DATA BYTE

C6E9 INCREMENT OFFSET (MAIN BUFFER)

C6E9 LOOP UNTIL WHOLE BUFFER IS DONE >>C69

C6EB *********** DETERMINE IF THERE IS MORE TO DO***********

C6EB INCREMENT MAIN BUFFER POINTER

C6ED INCREMENT SECTOR NUMBER

C6EF IS THERE ANOTHER SECTOR TO LOAD? ($00)

C6E6 YES, GO DO IT >>C6D3

C6EB NO, ENTER CODE WE JUST LOADED >>$01

C6F8 JUMP TO DRIVE 2 ENTRY POINT >>C6B

C6FE *********** UNUSED ***********

C6FE ---

C700 MAKE SLOT 7 LOOK EMPTY

C701 SELECT DEVICE 2

C703 SELECT DRIVE 2

C705 SELECT SLOT 6

C707 GO DO IT >>C6FB

---

Disk Controller Boot ROM -- Apple IIc

Next Object Addr: C6A6

Addr Description/Contents

C6A6 INITIALIZE OFFSET (AUXILIARY BUFFER)
Disk Controller Boot ROM -- Apple IIc

Next Object Addr: C707

<table>
<thead>
<tr>
<th>ADDR</th>
<th>Description/Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>C709</td>
<td>********** BUILD READ TRANSLATE TABLE **********</td>
</tr>
<tr>
<td>C709</td>
<td>INITIALIZE BIT PATTERN</td>
</tr>
<tr>
<td>C710</td>
<td>INITIALIZE TABLE VALUE INDICATOR</td>
</tr>
<tr>
<td>C710</td>
<td>STORE BIT PATTERN</td>
</tr>
<tr>
<td>C711</td>
<td>SHIFT PATTERN LEFT ONE BIT</td>
</tr>
<tr>
<td>C711</td>
<td>ARE THERE ANY TWO ADJACENT BITS ON?</td>
</tr>
<tr>
<td>C712</td>
<td>NO, TRY ANOTHER PATTERN &gt;&gt;C725</td>
</tr>
<tr>
<td>C713</td>
<td>YES, TURN OFF RIGHTMOST OF EACH GROUP OF ZEROS</td>
</tr>
<tr>
<td>C714</td>
<td>FLIP BITS, PAIR OF ZERO BITS NOW SINGLE BIT, ETC</td>
</tr>
<tr>
<td>C715</td>
<td>HIGH BIT ALWAYS ON/TURN OFF BIT WE MISSED BEFORE</td>
</tr>
<tr>
<td>C716</td>
<td>--- &gt;&gt;C725</td>
</tr>
<tr>
<td>C717</td>
<td>SHIFT PATTERN RIGHT, MUST HAVE ONLY ONE BIT ON</td>
</tr>
<tr>
<td>C717</td>
<td>IF MORE THAN ONE BIT ON, TRY ANOTHER PATTERN &gt;&gt;C71B</td>
</tr>
<tr>
<td>C718</td>
<td>FOUND ONE, GET TABLE VALUE</td>
</tr>
<tr>
<td>C719</td>
<td>AND STORE IT IN TABLE ($0356)</td>
</tr>
<tr>
<td>C720</td>
<td>INCREMENT TABLE VALUE INDICATOR</td>
</tr>
<tr>
<td>C720</td>
<td>GET NEXT BIT PATTERN, DONE YET?</td>
</tr>
<tr>
<td>C721</td>
<td>NO, GO CHECK IT OUT &gt;&gt;C70D</td>
</tr>
<tr>
<td>C721</td>
<td>MAIN BUFFER POINTER ($26) -&gt; $0800</td>
</tr>
<tr>
<td>C722</td>
<td>INITIALIZE RETRY COUNT (LOW BYTE)</td>
</tr>
<tr>
<td>C723</td>
<td>RETURN TO CALLER</td>
</tr>
<tr>
<td>C724</td>
<td>********** ASCII ERROR MESSAGE **********</td>
</tr>
<tr>
<td>C724</td>
<td>&quot;Check Disk Drive.&quot;</td>
</tr>
<tr>
<td>C740</td>
<td>TERMINATE STRING</td>
</tr>
</tbody>
</table>
Beneath Apple ProDOS Supplement

ProDOS Loader -- V1.0.1 -- 1 JAN 84
NEXT OBJECT ADDR: 0000

---

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>MODULE STARTING ADDRESS</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>PRODOS LOADER - LOADED FROM SECTORS</td>
</tr>
<tr>
<td></td>
<td>0 AND 2 OF TRACK 0 (BLOCK 0), LOADER</td>
</tr>
<tr>
<td></td>
<td>LOADS &quot;PRODOS&quot; FILE INTO MEMORY AT</td>
</tr>
<tr>
<td></td>
<td>AT $2000 AND BRANCHES TO IT.</td>
</tr>
<tr>
<td></td>
<td>(PRODOS RELOCATOR IS AT $2000)</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>VERSION 1.0.1 -- 1 JAN 84</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>***********************</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>0000</th>
<th>LOAD UP TO SET</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001</td>
<td>ON ENTRY</td>
</tr>
<tr>
<td></td>
<td>MAIN ENTRY</td>
</tr>
<tr>
<td></td>
<td>X = SLOT*16</td>
</tr>
<tr>
<td></td>
<td>A = SECTOR NUMBER</td>
</tr>
<tr>
<td>0001</td>
<td>ENTRY POINT</td>
</tr>
<tr>
<td>0002</td>
<td>ALWAYS TAKEN</td>
</tr>
<tr>
<td>0004</td>
<td>(NEVER EXECUT)</td>
</tr>
<tr>
<td>0007</td>
<td>SAVE SJ</td>
</tr>
<tr>
<td>0013</td>
<td>&gt;A132</td>
</tr>
<tr>
<td>0019</td>
<td>READING SECTO</td>
</tr>
<tr>
<td>002B</td>
<td>REMEMBER TH</td>
</tr>
<tr>
<td>002C</td>
<td>NEXT?</td>
</tr>
<tr>
<td>0030</td>
<td>MAKE SCS PROB</td>
</tr>
<tr>
<td>0052</td>
<td>AND SAVE AT $52</td>
</tr>
<tr>
<td>019F</td>
<td>$48/49 --&gt; 8X</td>
</tr>
<tr>
<td>019F</td>
<td>CHECK $CSFF</td>
</tr>
<tr>
<td>01DF</td>
<td>IN ROMBOOT</td>
</tr>
<tr>
<td>01F0</td>
<td>BOOT ROM.</td>
</tr>
<tr>
<td>0210</td>
<td>NO, HARD DISK II?</td>
</tr>
<tr>
<td>0211</td>
<td>GOT BOTH SECTS</td>
</tr>
<tr>
<td>0219</td>
<td>BOOL THEN &gt;&gt;00B</td>
</tr>
<tr>
<td>0230</td>
<td>NO, STOP AT SECTOR</td>
</tr>
<tr>
<td>0250</td>
<td>STOPE ON PARTIAL SECTOR</td>
</tr>
<tr>
<td>0253</td>
<td>IF (SECTOR 3)</td>
</tr>
<tr>
<td>02A2</td>
<td>DUMMY UP $CSFF</td>
</tr>
<tr>
<td>0300</td>
<td>GET SEC 2)</td>
</tr>
<tr>
<td>0389</td>
<td>AND CALL ROM.</td>
</tr>
<tr>
<td>0389</td>
<td>AS RETURN ADDRESS</td>
</tr>
<tr>
<td>038A</td>
<td>SECTOR READ SUBRTN</td>
</tr>
<tr>
<td></td>
<td>***********************</td>
</tr>
</tbody>
</table>

---

| 0048 | ADDRESS TO BLOCK READ ROUTINE |
| 004A | VOL DIR ENTRY POINT/ first INDEX PAGE |
| 004B | ADDR OF SECOND PAGE OF INDEX BLOCK |
| 0050 | INDEX INTO INDEX BLOCK PAGES |
| 0051 | TRACK SEEK PHASE-ON INDEX |
| 0052 | TRACK PHASE Wanted |
| 0053 | BLOCK READ ROCK COUNT |
| 0054 | CURRENT TRACK PHASE/PHASE-OFF INDEX |
| 0056 | BUFFER POINTER |
| 005A | SCREEN CENTER LINE |
| 2000 | LOAD POINT FOR RELOCATOR |
| 2000 | DISK ARM PHASES |
| 2000 | INHIBIT DISK DRIVE |
| 2000 | HOME CURSOR/CLEAR SCREEN |

---

| 0000 | LOAD PRODOS *************** |
|      | (ENTR IN MEMORY NOW) |
| 0033 | CURRENT TRACK |
| 0035 | $48/49 --> 8X IS ZERO |
| 0037 | COPY A PORTION |
| 0039 | TO MY BLOCK |
| 003D | FROM 09F2 TO |
| 003E | READER SUBROUTINE (0994) |
| 0040 | MODIFY SOME READS |
| 0046 | TO SUIT MY |
| 004A | BRANCHES IN THE CODED CODE (091D) |
| 004C | AND COPY SECTOR |
| 004F | HANDLING TASTES (0924) |
| 0055 | TO $48/49 --> 8X |
| 0057 | READ SUBROUTINE EXIT CODE (092B) |
| 0059 | AT $0986 |
| 005B | DISKETTE BLOCK READER SUBTRN |
| 005D | --- |
| 005F | HARD DISK ON |
| 0061 | DISKETTE? |
| 0063 | STORE ZEROS |
| 0065 | BLOCK READER |
| 006F | COMMAND = 1 |
| 0071 | IN SEVERAL THINGS |
| 0075 | $60/61 --> 5F |
| 0077 | $4A/4B --> 5C |
| 007C | (BUFFER) |
| 007C | (FIRST ENTRY) |
**ProDOS Loader -- V1.0.1 -- 1 JAN 84**

- **ADDR**
- **DESCRIPTION/CONTENTS**

**$0879** READ VOLUME DIRECTORY BLOCKS $0912

**$087C** ERROR? >>$08E6

**$087E** BUMP TO NEXT BLOCK (2 PAGES)

**$0882** NEXT BLOCK NUMBER

**$0886** NOW AT BLOCK 67

**$0888** NO, GO READ NEXT ONE >>$0879

**$0889** YES, CHECK LINK FOR VALIDITY ($0C00)

**$088D** IT SHOULD BE ZERO FOR VOL DIR ($0C01)

**$0890** WASTY VOL DIR? >>$08FF

**$0892** NO, INDEX PAST LINK AND VOL HDR

**$0894** AND BEGIN >>$0898

**$0896** IF ALREADY PROCESSING, USE ENTRY LSB

**$0898**

**$0899** ADD ENTRY LENGTH TO FIND NEXT ENTRY ($0C23)

**$089D** STILL IN SAME PAGE? >>$08AC

**$089F** NO, BUMP ENTRY MSB

**$08A3** IS IT ODD? (SECOND PAGE OF A BLOCK?)

**$08A4** YES... >>$08AC

**$08A6** NO, JUST FINISHED LAST BLOCK?

**$08A8** YES, ERROR -- FILE NOT FOUND >>$08FF

**$08A9** ELSE, START JUST PAST LINKS

**$08AC** UPDATE LSB OF ENTRY POINTER

**$08AE** GET NAME LENGTH ($0902)

**$08B1** TURN OFF FLAGS

**$08B4** COMPARE NAME WITH "PRODOS"

**$08B9** NOT A MATCH? >>$0896

**$08BE** IF NAME MATCHES, IS IT A SAPLING FILE?

**$08C2** IF NOT, I CAN'T HANDLE IT >>$08FF

**$08C6** GET FILE TYPE

**$08CB** SHOULD BE A PRODOS SYS FILE

**$08CA** IF NOT, I GIVE UP >>$08FF

**$08CD** ALL IS WELL, COPY KEY BLOCK NUMBER

**$08CF** TO $46/$47

**$08D6** $4A/$4B AND $6A/$61 --> $1800

**$08DB** (BUFFER TO HOLD KEY BLOCK)

**$08E1** $4C/$4D --> $1F00 (SECOND PAGE)

**$08E3** READ A BLOCK <$0912>

**$08E6** ERROR? >>$08FF

**$08EA** BUMP TO NEXT BLOCK BUFFER

**$08EE** $4E = OFFSET INTO INDEX BLOCK

**$08F0** GET NEXT BLOCK NUMBER FROM INDEX BLOCK

**$08FB** BLOCK NUMBER -- $2 (END OF FILE)

**$08FA** NOT YET, READ A BLOCK >>$08E3

**$08FC** ELSE, JUMP TO RELOCATOR AT $2000 >>$2000

**$08FF** ERROR JUMP >>$093F

**$0902**

- **DESCRIPTION/CONTENTS**

**$0902**

**$0903**

**$0912** 

- **KERNEL NAME**

**$0912**

- **NAME OF KERNEL'S NAME**

**$0914**

- **PRODOS**

**$091A**

- **COPY BLOCK READ BUFFER PTR**

**$091D**

- **Y $68/$61 --> $44/$45

- **LOCK READ BUFFER POINTER**

- **I GO TO BLOCK I/O ROUTINE >>$048**

**$091D**

- **ROM SECTOR READ OFFSETS**

**$091D**

- **OFFSETS INTO ROM SECTOR READ SUBROUTINE TO BRANCH DISPLACEMENTS WHICH NEED TO BE CHANGED FOR LOADER'S PURPOSES**

**$0924**

**$092B**

- **NEW BRANCH OFFSETS FOR ABOVE**

**$092B**

**$092D**

- **SECTOR READ EXIT CODE**

**$092F**

- **COPIED INTO DISKETTE SECTOR READ CODE**

**$0932**

- **S0**

**$0932**

- **D EXIT NORMALLY**

**$0932**

- **START BLOCK READ OPERATION >>$093C**

**$093F**

- **932-93E NOT USED**

**$093F**

**$0943**

- **ERROR HANDLER**

**$0947**

- **CLEAR CURSOR/SCREEN <C58>**

**$0950**

- **"UNABLE TO LOAD PRODOS" MESSAGE ($05AE)**

**$0950**

- **I GO TO SLEEP FOREVER >>$094D**

**$096D**

- **UNABLE TO LOAD PRODOS***

**$0972**

- **MOVE ARM TO NEXT PHASE***

**$0975**

- **CURRENT PHASE**

**$097A**

- **VERT TO NEXT ARM PHASE**

**$0980**

- **NEXT ARM PHASE THIS DRIVE**

---

Beneath Apple ProDOS Supplement
Beneath Apple ProDOS Supplement

ProDOS Loader -- V1.0.1 -- 1 JAN 84

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>097C</td>
<td>DELAY LONG ENOUGH TO MOVE DISKETTE BLOCK ARM TO MOVE</td>
</tr>
<tr>
<td>0983</td>
<td>WHEN FINISHED, RETURN</td>
</tr>
<tr>
<td>0985</td>
<td>RETURN</td>
</tr>
<tr>
<td>0986</td>
<td>DISKETTE BLOCK ARM TO MOVE</td>
</tr>
<tr>
<td>0987</td>
<td>44/45 --&gt; BUFFER</td>
</tr>
<tr>
<td>0988</td>
<td>46/47 = BLOCK NO.</td>
</tr>
<tr>
<td>0989</td>
<td>READ BLOCK NO. LSB</td>
</tr>
<tr>
<td>098A</td>
<td>READ ROUTINE START OF SECTOR READ ROUTINE</td>
</tr>
<tr>
<td>098B</td>
<td>DISKETTE BLOCK ARM TO MOVE</td>
</tr>
<tr>
<td>098C</td>
<td>ISOLATE SECTOR REMAINDER</td>
</tr>
<tr>
<td>098D</td>
<td>SKEW SECTOR BY 2</td>
</tr>
<tr>
<td>098E</td>
<td>AND STORE SECTOR VALUE</td>
</tr>
<tr>
<td>098F</td>
<td>GET MSB</td>
</tr>
<tr>
<td>0990</td>
<td>HIGH BIT OF TRACK</td>
</tr>
<tr>
<td>0991</td>
<td>MERGE WITH LOW PART</td>
</tr>
<tr>
<td>0992</td>
<td>STORE TRACK WANTED</td>
</tr>
<tr>
<td>0993</td>
<td>TRACK*2 IS PHASE W</td>
</tr>
<tr>
<td>0994</td>
<td>SET PAGE ADDRESS OF</td>
</tr>
<tr>
<td>0995</td>
<td>X</td>
</tr>
<tr>
<td>0996</td>
<td>TURN DRIVE MOTOR ON</td>
</tr>
<tr>
<td>0997</td>
<td>OF TRACK</td>
</tr>
<tr>
<td>0998</td>
<td>READ SECTOR &lt;9BC&gt;</td>
</tr>
<tr>
<td>0999</td>
<td>NEXT PAGE</td>
</tr>
<tr>
<td>099A</td>
<td>SKEW TO NEXT SECTOR</td>
</tr>
<tr>
<td>099B</td>
<td>READ SECOND SECTOR</td>
</tr>
<tr>
<td>099C</td>
<td>THEN TURN MOTOR OFF</td>
</tr>
<tr>
<td>099D</td>
<td>RETURN</td>
</tr>
<tr>
<td>099E</td>
<td>DISKETTE BLOCK ARM TO MOVE</td>
</tr>
<tr>
<td>099F</td>
<td>AND EXIT (C068)</td>
</tr>
</tbody>
</table>

ProDOS Loader -- V1.0.1 -- 1 JAN 84

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>09F2</td>
<td>START OF SECTOR READ ROUTINE</td>
</tr>
<tr>
<td>09F3</td>
<td>BASE ADDR FOR MODIFICATIONS</td>
</tr>
<tr>
<td>09F4</td>
<td>A86-BFF NOT USED</td>
</tr>
<tr>
<td>09F5</td>
<td>DISKETTE BLOCK ARM TO MOVE</td>
</tr>
<tr>
<td>09F6</td>
<td>DEVICE DEPENDENT SECTOR READ ROUTINE</td>
</tr>
<tr>
<td>09F7</td>
<td>COPIED FROM ROM ON DISKETTE CARD</td>
</tr>
<tr>
<td>09F8</td>
<td>SEE SCX5E IN BOOT ROM</td>
</tr>
<tr>
<td>09F9</td>
<td>VOLUME DIRECTORY BUFFER</td>
</tr>
<tr>
<td>09FA</td>
<td>START OF VOLUME DIRECTORY BUFFER</td>
</tr>
<tr>
<td>09FB</td>
<td>OFFSET TO ENTRY LENGTH FIELD</td>
</tr>
<tr>
<td>09FC</td>
<td>----</td>
</tr>
<tr>
<td>09FD</td>
<td>----</td>
</tr>
<tr>
<td>09FE</td>
<td>----</td>
</tr>
<tr>
<td>09FF</td>
<td>----</td>
</tr>
</tbody>
</table>
Beneath Apple ProDOS Supplement

---

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td>MODULE STARTING ADDRESS</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>*************************</td>
</tr>
<tr>
<td></td>
<td>* PRODOS RELOCATOR *</td>
</tr>
<tr>
<td></td>
<td>* LOADED AS THE FIRST *</td>
</tr>
<tr>
<td></td>
<td>* PORTION OF THE PRODOS *</td>
</tr>
<tr>
<td></td>
<td>* IMAGE AT $2000. *</td>
</tr>
<tr>
<td></td>
<td>* VERSION 1.0.1 -- 1 JAN 84 *</td>
</tr>
<tr>
<td></td>
<td>*************************</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>***********************</td>
</tr>
<tr>
<td></td>
<td>** SCREEN LINE ADDRESSES *******</td>
</tr>
<tr>
<td></td>
<td>04B6 SCREEN BUFFER LINE</td>
</tr>
<tr>
<td></td>
<td>05A9 SCREEN BUFFER LINE</td>
</tr>
<tr>
<td></td>
<td>05AE SCREEN BUFFER LINE</td>
</tr>
<tr>
<td></td>
<td>06A3 SCREEN BUFFER LINE</td>
</tr>
<tr>
<td></td>
<td>07A8 SCREEN BUFFER LINE</td>
</tr>
<tr>
<td></td>
<td>07AD SCREEN BUFFER LINE</td>
</tr>
<tr>
<td></td>
<td>07D0 SCREEN BUFFER LINE</td>
</tr>
<tr>
<td></td>
<td>***********************</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>***********************</td>
</tr>
<tr>
<td></td>
<td>** INTERP LOADER ADDRESSES *******</td>
</tr>
<tr>
<td></td>
<td>0800 ENTRY OF INTERP LOADER</td>
</tr>
<tr>
<td></td>
<td>08E2 'UNABLE TO FIND SYSTEM FILE'</td>
</tr>
<tr>
<td></td>
<td>090A 'INTERP FILE TOO LARGE'</td>
</tr>
<tr>
<td></td>
<td>092A 'UNABLE TO LOAD ...'</td>
</tr>
<tr>
<td></td>
<td>093B INTERP FILE NAME ITSELF</td>
</tr>
<tr>
<td></td>
<td>093C +1</td>
</tr>
<tr>
<td></td>
<td>094F LENGTH OF MESSAGE</td>
</tr>
<tr>
<td></td>
<td>0950 MLI: OPEN LIST</td>
</tr>
<tr>
<td></td>
<td>0956 MLI: GET EOF</td>
</tr>
<tr>
<td></td>
<td>0958 EOF MARK</td>
</tr>
<tr>
<td></td>
<td>0959 EOF MARK+1</td>
</tr>
<tr>
<td></td>
<td>095A EOF MARK+2 (MSB)</td>
</tr>
<tr>
<td></td>
<td>095B MLI: READ LIST</td>
</tr>
<tr>
<td></td>
<td>095C READ BUFFER ADDR</td>
</tr>
<tr>
<td></td>
<td>0960 +1</td>
</tr>
<tr>
<td></td>
<td>0963 MLI: CLOSE LIST</td>
</tr>
<tr>
<td></td>
<td>0965 '.SYSTEM'</td>
</tr>
<tr>
<td></td>
<td>0CA0 VOLUME DIRECTORY BUFFER</td>
</tr>
<tr>
<td></td>
<td>0C23 ENTRY LENGTH</td>
</tr>
<tr>
<td></td>
<td>-- RAMDRIVE VOLUME DIRECTORY</td>
</tr>
<tr>
<td></td>
<td>0E04 VOLUME HDR, VOLUME NAME</td>
</tr>
<tr>
<td></td>
<td>0E22 VOLUME HDR, ACCESS-TOTAL BLOCKS</td>
</tr>
<tr>
<td></td>
<td>***********************</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>***********************</td>
</tr>
<tr>
<td></td>
<td>** SYSTEM GLOBAL PAGE *******</td>
</tr>
<tr>
<td></td>
<td>BF00 ENTRY POINT FOR MLI</td>
</tr>
<tr>
<td></td>
<td>BF03 QUIT VECTOR</td>
</tr>
<tr>
<td></td>
<td>BF06 DATE/TIME</td>
</tr>
<tr>
<td></td>
<td>BF10 DEVICE HANDLER TABLES</td>
</tr>
<tr>
<td></td>
<td>BF30 LAST DEVICE USED</td>
</tr>
<tr>
<td></td>
<td>BF31 NUMBER OF ACTIVE DISK DEVICES</td>
</tr>
<tr>
<td></td>
<td>BF32 ACTIVE DISKS SEARCH LIST</td>
</tr>
<tr>
<td></td>
<td>BF98 MACHINE TYPE FLAGS</td>
</tr>
<tr>
<td></td>
<td>BF99 SLOT WHICH CONTAIN CARDS WITH ROM</td>
</tr>
<tr>
<td></td>
<td>BFNN TOP OF 48K RAM</td>
</tr>
</tbody>
</table>

---

ProDOS Relocator -- V1.0.1 -- 1 JAN 84
NEXD OBJECT ADDR: 2000

---

PRODOS Relocator -- V1.0.1 -- 1 JAN 84
NEXT OBJECT ADDR: 2000

---
ProDOS Relocator

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

C000 80 STORE OFF
C001 80 STORE ON
C002 READ MAIN RAM
C003 READ AUX RAM
C004 WRITE MAIN RAM
C005 WRITE AUX RAM
C008 MAIN STACK/EMS
C009 ALTERNATE EMU
C00C 88 COLUMN DISPLAY
C018 READ 80 STOR DISPLAY
C030 SPEAKER SWIT
C082 MOTHERBOARD
C083 READ/COMPLETE
C08B READ/WRITE 2ND LINES
C311 MOVE TO/FROM ACME
C314 XFER TO/ACME
CF8F RESET 1/O CIRCUIT

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

D000 KERNEL START
FF00 START OF DEF ARP

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

FB1E PADDLE READ
FB2F MONITOR INI SUBROUT
FBB3 ROM VERSION: ROUT
FC00 SECONDARY $WYTE
FC58 CLEAR SCREEN
FE84 SET NORMAL
FE89 IN# VIDEO
FE93 PRO

2000 STORE SLOT IN 15
2005 PRINT "APPLE..."
200E MOVE $PAGE:111
211 NO ERROR? >INTERR
213 ERROR >>212/216
216 ---
21A THERE MUST BE...
221 IF NOT, ERROR A
229 MAKE DOUBLY >>24
22B SELECT MOTHERBOARD

ProDOS Relocator -- V1.0.1 -- 1 JAN 84

**ADD** DESCRIPTION/CONTENTS

20CB 128K?
20CF NO...>>20D4
20D1 YES, ESTABLISH RAM DRIVE IN UPPER 64K <28FF>

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

20D4 MLI: ONLINE DEVICE CALL <BF00>
20DA ERROR? >>212A /
20DF VALID VOLUME NAME?
20E1 IF NOT, ERROR >>212A
20E4 ELSE, BUMP LENGTH BY ONE
20E9 AND PREFIX NAME BY A "/"
20EE MLI: SET PREFIX <BF00>
20F4 ERROR? >>212A

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

20F6 ---
20F8 14/15 --> $C00
20FE ---
2103 BLOCK = 2 (VOLUME DIRECTORY) (216F)
2109 MLI: READ BLOCK <BF00>
210F ERROR? >>212A
2113 GET NEXT BLOCK NUMBER
2119 IF ZERO, END OF VOLUME DIRECTORY >>2127
2121 ADD TWO PAGES (ONE BLOCK) TO POINTER
2123 AND STOP AT $1400 IN ANY CASE
2125 ELSE, READ NEXT BLOCK AS WELL >>20FE
2127 WHEN DONE, JUMP TO INTERP LOADER >>$000

212A ********* ERROR HANDLER *****************

212A ENABLE MOTHERBOARD ROMS (C082)
212D CLEAR SCREEN <FC50>
2132 PRINT "RELOCATION/CONFIG ERROR" (213E)
213B THEN SLEEP FOREVER >>213B

213E ********* DATA **************************

213E ---

RELOCATION / CONFIGURATION ERROR ***

2164 MLI: ONLINE PARGS
2165 SLOT*16 AND DRIVE
2166 READ THEM TO $281

2168 MLI: SET PFX PARGS
2169 PREFIX IS AT $280

ProDOS Relocator -- V1.0.1 -- 1 JAN 84

**ADD** DESCRIPTION/CONTENTS

216B MLI: READ BLOCK PARMS
216C DEVICE
216D BUFFER
216F BLOCK NUMBER

2171 ADDRESSES OF RELOCATION TABLES

217D ********** RELOCATION TABLES ***********************

+0: $0 - ZERO BLOCK OF MEMORY
+1: COPY BLOCK
+2 - RELOCATE MSB ADDRESSES
+3: RELOCATE 2 BYTE ADDR
+4: RELOCATE INSTRUCTIONS
+1/2: ADDR OF OUTPUT BLOCK
+3/4: LENGTH OF BLOCK IN BYTES
+5/6: ADDR OF INPUT BLOCK (IF ANY)
+7: NUM RANGES TO CORRECT FOR (-1)
+8: START PAGES
+8+COUNT: END PAGE ADDRESSES
+8+COUNT+COUNT: ADDITIVE CORRECTION FACTOR

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

217D COPY (INTERPRETER LOADER)
217E TO =$900
2180 LKN=$16C
2182 FRM=$2234
2184 COPY (3 PAGE IMAGE)
2185 TO =$3F0
2187 LKN=$10
2189 FRM=$23A0
218B COPY (CHECKSUM)
218C TO =$8A
218E LKN=$982
2190 FRM=$14
2192 COPY (RAM DRIVE BANK SWITCHER)
2193 TO =$80
2195 LKN=$47
2197 FRM=$2401
2199 END OF TABLE

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

219A COPY (QUIT CODE)
219B TO =$0100
219D LKN=$100
219F FRM=$5900
21A1 END OF TABLE
Beneath Apple ProDOS Supplement

ProDOS Relocator -- V1.0.1
ADDRESS DESCRIPTION DATA AREA

******** 4K PRODOS

21A2 COPY (SYSTEM GLOBAL)
21A3 TO = D8F00
21A5 LEN = 5100
21A7 FRM = 54F00
21A9 ZERO (PRODOS KERNEL)
21AA ADR = 8100
21AC LEN = 5700
21AC COPY (PRODOS KERNEL)
21AF TO = 59000
21B1 LEN = 52100
21B3 FRM = 52000
21B5 RELOCATE INSTRUCTION
21B6 TO = 59000
21B8 LEN = 51EC
21BA Fx = 59000
21BC FOR ADDR = 0
21BF ADJUST BY = 0
21C0 RELOCATE ADDRESSES
21C1 TO = 5AF65
21C3 LEN = 528
21C5 Fx = 5AF65
21C7 FOR ADDR = 5F8XX
21CA ADJUST BY = 5C0
21CB COPY (DEVICE DRIVERS)
21CC TO = 8000
21CE LEN = 5700
21DB Fx = 52000
21D2 RELOCATE INSTRUCTION
21D3 TO = 8000
21D5 LEN = 195
21D7 Fx = 5800
21DC FOR ADDR = 0
21DD ADJUST BY = 0
21DE TO = 6339
21E0 LEN = 6339
21E2 Fx = 6339
21E4 FOR ADDR = LEN
21E7 ADJUST BY = 0
21E8 END OF TABLE

******** 4K PRODOS

21E9 COPY (CLOCK DRIVER)
21EA TO = 8142
21EC LEN = 5700
21EE FRM = 55000
21F0 RELOCATE INSTRUCTION

ProDOS Relocator -- V1.0.1
ADDRESS DESCRIPTION DATA AREA

21F1 TO
21F3 LEN = 5K PRODOS CLOCK TABLE
21F5 FRM
21F7 FOR (CODE)
21FA ADDRES
21FE END OF TABLE

******** 4K PRODOS

21FF COPY (INTER )
2200 TO = 8100
2202 LEN = 5280
2204 LEN = 580
2206 COPY (SYSTEM)
2207 TO
2209 LEN
220B FRM
220D ZERO (PRODOS)
220E ADDRES
2210 LEN
2212 COPY (PRODOS)
2213 TO
2215 LEN
2217 FRM
2219 COPY (DEVICE)
221A TO
221C LEN
221E FRM
2220 END OF TABLE

********

2221 COPY (CLOCK)
2222 TO
2224 LEN
2226 FRM
2228 RELOCATE INT
2229 TO
222B LEN + 0.1 -- 1 JAN 84
222D FOR CONTENTS
222F FOR CONTENTS
2232 ADDR
2233 END OF TABLE

******** 4K PRODOS RELOC TABLE

2342 TO
2344 ADDRES=SC1-C1
2346 BY = 580 AND 5C0
2348 PRODOS RELOC TABLE

(UP T VECTORS)
2350 ""
2352 ""
2354 "GLOBAL PAGE"
2356 ""
2358 "" (KERNEL DATA AREA)
235A "" (KERNEL)
235C "" (GLOBAL PAGE)
22A6 ERROR? >>22EE
22A8 MLI: GETOF <BF00>
22AC (PARM LIST AT $238A)
22AE ERROR? >>22EE
22B0 GET MSB (SEE $238E) (095A)
22B3 BIGGER THAN 64K??? >>2308
22B8 MUST BE LESS THAN 98000 BYTES
22BA OR ERROR... >>230B
22BC STORE LENGTH IN MLI READ LIST (0964)
22C2 AND LSB TOO (095F)
22C5 MLI: READ INTERPRETER INTO $20008 <<
22C9 (PARM LIST AT $238F)
22CB NO ERRORS? >>22D3
22CD ERROR, BAD BUFFER?
22CF YES, FILE WAS TOO LARGE >>2308
22D1 ELSE, "UNABLE TO LOAD ..." >>22EE
22D3 MLI: CLOSE INTERPRETER FILE <BF00> 90A)
22D7 (PARM LIST AT $2397)
22DA ERROR? >>22EE
22DB NO, ENABLE MOTHERBOARD ROMS (C9?) ******* SYSTEM FILE **
22DE AND JUMP TO INTERPRETER >>2000

22E1 ******* ERROR HANDLERS ************** LARGE ***

22E1 ---
22E3 PRINT "UNABLE TO FIND A .SYSTEM P"
22EC THEN GO TO SLEEP >>2313

22EE GET NAME LENGTH (094F)
22F1 LINE LENGTH
22F4 LESS NAME LENGTH (094F)
22F7 DIVIDED BY 2
22F8 GIVES OFFSET TO CENTER THE LINE (09)
22FA PRINT "UNABLE TO LOAD ..." (092A)
2306 GO TO SLEEP FOREVER >>2313
Beneath Apple ProDOS Supplement

ProDOS Relocator -- V1.01

---------- DESCRIPTION/CONTENT ----------

2384 MLI: OPEN PARM LIST
2385 PATHNAME IS AT $2800:
2387 I/O BUFFER AT $1400
2389 REFNUM=1

238A MLI: GET EOF PARM LST
238B REFNUM=1
238C EOF MARK POSITION

238F MLI: READ LIST
2390 REFNUM=1
2391 READ TO $2800
2393 LENGTH (FROM EOF MARK)
2395 ACTUAL LENGTH READ

2397 MLI: CLOSE LIST
2398 REFNUM=0, CLOSE ALL

2399 '.SYSTEM'

23A0 ********** END OF INT.

23A8 *********** 3 PAGE VEO

23A9 BRK HANDLER AT $PA:
23A2 RESET AT $FF59
23A4 POWER UP BYTE
23A5 & VECTOR TO $FF59
23A8 CTL-Y VECTOR TO $FF
23AB NMI VECTOR TO $FF59
23AE IRQ HANDLER AT $FF59

23B0 ********** DETERMINE 59 IMAGE **************

23B0 SUC=80...
23B1 01...
23B2 11...
23B3 01 MAC59
23B4 10 >>PF59
23B5 11 >>PF59

23B6 MACHINE ID ***************

23B8 I. APPLE II
23B9 3. APPLE II+
23BA 1. APPLE IIE
23BB . APPLE // EMULAT.
23BC 48K RAM
23BD 128K RAM
23BE CURRENT MACHINE
23BF FUTURE MACHINE
23C0 80 COL CARD
23C1 THUNDER CLOCK

2401 **********

2401 MAKER

---------- PRODOS Relocator ----------

---------- ADDRESS DESCRIPTION/CONTENT ----------

23B0 ASSUMED TO 880 (CODE BUSY)
23B4 GET A
23B5 APPLE ** LOOK FOR EXT
23B7 YES, 82 HAVED TO 880 TO
23B9 NO,
23BB APPLE & MACHID
23BF YES, C>243
23CA NO,
23CC APPLE NO AUX MEMORY, (CO) NO
23CD NO, NOT A PATTERN THERE
23CE REALMORD PATTERN STAY
23CC YES >
23D0 // BT, ONLY ONE 64K TOTAL
23D2 YES * GOT 124K TOTAL
23D4 OTHERWISE DO MAIN MEMORY
23D6 CREAT>268
23D8 AND GO b: 128K
23DA UPDATE C
23DB READ/Y SA/XA >> *APPLE
23DC CHECK IF START MONITOR 2
23DE IF PRB=9
23FA ELSE, DO A CONVOLUTE
23FC ADD THE 50 CALLER
23FE THEN =

2401 **********

2401 UPDATE

2403 IF E7

2405 YES,

2407 BANK

240D STORE

2416 MAKE

2420 ---

2422 IF NOT

2424 ELSE:

2426 BANK

2432 64K?

2436 NO, 1:

2438 IN MAN

243A SET UI

243D IN ADI

243F AT SF

2441 BUT D

2447 RETURN
### ProDOS Relocator -- V1.0.1 -- 1 JAN 84

#### ADDR | DESCRIPTION/CONTENTS
--- | ---
2448 **DISPLAY LOAD MESSAGE**
244B CLICK SPEAKER (C030)
244D STORE IN MAIN MEMORY (C00C)
244E 80 COL DISPLAY OFF (C008)
2451 SET NORMAL VIDEO <FE84>
2454 CALL MONITOR INITIALIZATION <FB2F>
2457 SET VIDEO PR40 <FE93>
245A SET KEYBD INH (FE89)
245D OUT OF DECIMAL MODE
245E DISABLE FOR INTERRUPTS
245F CLEAR SCREEN <FC58>
2464 PRINT "APPLE II" (2492)
246F PRINT "PRODOS 1.0.1 ETC." (249A)
247A PRINT BLANKS (24B1)
2485 PRINT "COPYRIGHT ETC." (24BF)
248E CLICK SPEAKER AGAIN (C030)
2491 DONE

#### DATA AREA

2492 'APPLE II'
249A 'PRODOS 1.0.1 1-JAN-84'
24BF 'COPYRIGHT APPLE COMPUTER, INC., 1983-84'

#### DETERMINE SLOT CONFIGURATION

24E6 ---
24E8 ZERO SOME THINGS
24EF NO DISKS ACTIVE YET (BF31)
24F4 $10/11 --> SC700 (LOOP THRU ALL SLOTS)
24F6 RESET I/O CARD ROMS (CF0F)
24FB CHECK SIGNATURE ON CARD FOR DISK DEVICE
2500 NOT DISK? >>2569
2507 GET SCSFP BYTE (TYPE OF DISK)
2509 DISK II? >>252B
250B NO, PROFILE?
250D NO? THEN NOT A DISK >>2569

**PROFILE FOUND**

250F ELSE, SAVE AS LSB OF BLOCK READ SUBRTN
2511 GET SCSFP (STATUS BYTE)
2514 CAN WE AT LEAST READ STATUS AND DATA?
2518 YES? >>251F
251A NO,
251D NOT A DISK AFTER ALL >>2569
251F GET STATUS BYTE AGAIN

#### ProDOS Relocator -- V1.0.1 -- 1 JAN 84

#### ADDR | DESCRIPTION/CONTENTS
--- | ---
2523 TOP NIBBLE IS DEVICE ID
2524 PROFILE SHOULD BE $04
2526 CHECK NUMBER OF VOLS (SHOULD BE $0)
2527 GET SLOT NO. FOR DEVICE DRIVER LOC.
2529 AND GO DO COMMON PROCESSING FOR DISK >>2535

**DISK II FOUND**

252B $12 ZERO FOR DISK II
252D GET DISK II DEVICE DRIVER LOCATION (2627)
2531 ($F800 OR $B800) (2628)
2534 DISK II HAS 2 DRIVES

**DISK FOUND**

2535 SAVE DEVICE ADDRESS
2537 SET UP INDEX OF SLOT*2
253F BUILD ST (S=SLOT, T=O DISKII,4 PROFILE)
2542 BUMP DEVICE COUNT BY ONE (BF31)
2546 AND ADD DRIVE TO SYSTEM SEARCH LIST (BF32)
254A NUMBER OF DRIVES
254C ONLY ONE? >>2552
254E NO, BUMP INDEX
254F AND MARK SECOND DRIVE IN SEARCH LIST (BF32)
25503 STORE FINAL DEVICE COUNT (BF31)
25507 SET UP DEVICE DRIVER VECTORS (BF11)
255A IN SYSTEM GLOBAL PAGE >>2564
255C (SET UP TWO VECTORS FOR A DISK II) (BF21)
2564 ---
2568 I RECOGNIZE THIS CARD
2569 GO MARK SLTRY TO SHOW ROMS IN SLOT <25B6>
2570 DO ALL CARDS EXCEPT
2572 SLOT 0 ($C000) >>24F6
2578 GET LAST DISK DEVICE IN SEARCH LIST (BF32)
257B BOOT DRIVE? (BF30)
257B NO, KEEP LOOKING >>2566
2586 ---
2589 GET DEVICE COUNT (BF31)
258D IS BOOT DRIVE IN LIST? >>25A3
258F SO IT WILL BE SEARCHED FIRST... (BF30)
2592 STORE BOOT AT END OF SEARCH LIST (BF32)
2596 ANY OTHERS? >>25AA
2599 YES, SECOND DRIVE? >>25A3
259D STORE IT RIGHT BEHIND BOOT DRIVE (BF32)
25A1 NOW ANY MORE? >>25AA
25A3 ---
25AA YES, MOVE OTHERS AHEAD IN LIST (BF32)
25AA DO CHECKSUM ON ROM <2639>
25AD NOT AN AUTOSTART ROM? >>25B3
25AF AUTOSTART, STORE FINISHED MACHID (BF98)
25B2 AND LEAVE
Beneath Apple ProDOS Sup

ProDOS Relocator -- VI

-- 1 JAN 84

NEXT OBJECT ADDR: 25B3

ADDRES DESCRIPTION/CONTENTS

25B3 NONAUTOSTART, 1

25B6 IDENT

25B6 DO WE ALREADY KNOW MACHINE? >>23D4

25B8 NO,

25BA CHECK SIGNATURE

25BF NOT IT? >>25B6

25C5 THUNDER CLOCK WAHICH SLOT?

25C9 SAVE SLOT NUMBER

25CC IN CLOCK CODE REFERENCE TABLE (2232)

25D1 ENABLE CLOCK/CLOCK JUMP IN GLOBALS (BF96)

25D6 IS THERE A MACHINE CLOCK IS PRESENT

25DA AND UPDATE MACH

25DC GO MARK ROM IN THIS SLOT >>2618

25DE =-------

25E0 CHECK SIGNATURE M BY MISTERY CARD?

25E2 STANDARD BASIC >>2607

25E4 NO, UNKNOWN CARD

25E6 YES,

25E8 DOUBLE CHECK BASIC >>2607

25EC NO, UNKNOWN CARD

25F0 YES,

25F2 GENERIC SIGNATURE >>2607

25F4 NO, UNKNOWN CARD

25F7 YES,

25F9 80 COLUMN CARD?

25FB NO, UNKNOWN CARD >>2607

25FF GET MACHID IF CARD PRESENT

2601 MARK 80 COLUMN

2603 AND UPDATE MACH

2605 GO MARK ROM ON CLOCK ROM TO...

2607 UNKNOWN CARD, NO VALUE...

260B SEE IF IT WILL

2611 FOR SOME TIME.

2618 IF SO, WE HAVE A CARD IN SLOT

261A CONVERSION >>2619

261D TO A BIT POSITION >>261E

2620 AND OR INTO Slot (BF99)

2626 RETURN TO CAL

2627 DATA

2627 DISK DEVICE DR

2628 ENTRY POINT

2628 (2 BYTE ADDRESS)
ProDOS Relocator -- V1.0.1 -- 1 JAN 84  
NEXT OBJECT ADDR: 26D1

---

26D1  RESTORE OPERATION CODE (283C)
26D6  RELOCATE INSTRUCTIONS? >>26D6

26D8  ********** 2/3 - RELOCATE ADDRESSES  ***************

26DB  NO, RELOCATE ADDRESS <277A>
26DC  COPY BLOCK <2723>
26DE  AND CONTINUE IF ALL WENT WELL >>2667
26E1  NORMAL EXIT
26E2  RETURN
26E3  JUMP TO ERROR EXIT >>27B0

26E6  ********** 4 - RELOCATE INSTRUCTIONS  ***************

26E9  RELOCATE INSTRUCTIONS <278C>
26E9  AND THEN COPY BLOCK >>26DB

26EC  ********** 0 - ZERO BLOCK  ***************

26F1  BUMP TABLE POINTER TO NEXT ENTRY <2716>
26F6  GET NUMBER OF PAGES TO DO
26F6  BUMP PAGE POINTER
26F7  AND DECMENT LENGTH
2701  GET LENGTH OF PARTIAL LAST PAGE
2703  NO PARTIAL PAGE? >>270D
2706  ZERO PARTIAL PAGE TOO
270D  DONE, GET NEXT TABLE ENTRY >>2667

271B  ********** 1 - COPY BLOCK  ***************

271B  BUMP TABLE POINTER <2716>
271E  AND GO COPY BLOCK >>26DB

271E  ********** ADVANCE TABLE POINTER  ***************

2716  ADD FINAL ENTRY INDEX...
271A  TO TABLE ENTRY ADDRESS
2722  RETURN

2723  ********** COPY BLOCK  ***************

2723  ----
2727  INPTR < OUTPTR? >>2734
2729  NO, GREATER? >>2757
272B  MSB'S ARE EQUAL, CHECK LSB'S ALSO
2733  EXIT IF EQUAL
2734  INPTR < OUTPTR, COPY LAST PAGES FIRST
2738  BUMP BOTH INPTR AND OUTPTR BY...
273A  LENGTH-1 TO POINT AT LAST BYTE

ProDOS Relocator -- V1.0.1 -- 1 JAN 84  
NEXT OBJECT ADDR: 2742

---

2742  START WITH SHORT LAS
2746  ----
2747  GET PAGE LENGTH
2747  COPY BYTES BACKWARDS
2748  DROP ADDRESSES AND: THROUGH MEMORY
2754  AND CONTINUE UNTIL LENGTH BY 256
2756  RETURN

2757  INPTR > OUTPTR, COPY
2759  HOW MANY FULL PAGES ? PAGES FORWARD
275B  NONE? >>276C ? LEFT?
275D  COPY A FULL PAGE
275E  AND BUMP ADDRESSES
2768  DECMENT LENGTH BY 1
276A  AND DO ALL PAGES >>2356
276C  GET LENGTH OF LAST PAGE
276E  EVEN PAGE BOUNDARY
2770  NO, COPY SHORT LAST >>2779
2779  RETURN

277A  ********** ADDR/PAGE DESCRIPTION/RELOCATE  **********

277A  GET TABLE ENTRY TYPE
277C  GET PAGE TO RELOCATE (283C)
2780  RELOCATE A SINGLE ACK
2783  BUMP BY 1 OR 2 BYTES DRESS <27BB>
2786  ADVANCE POINTER <27D0 (283C)
2789  AND CONTINUE UNTIL C?
278B  RETURN

278C  ********** INSTRUCTIONS  **********

278C  ---
278E  GET 6582 OPCODE
2790  COMPUTE INSTRUCTION ?
2793  INVALID OPCODE? >>27E7 LENGTH <27E7>
2795  3 BYTE INSTRUCTIONS</E6
2797  AND BUMP BY 2 BYTES
2799  AND ADVANCE BY 3 BYTES
27A0  NEXT INSTRUCTION <2748>
27A3  CONTINUE UNTIL FIND34
27A5  RETURN

27A6  ********** INVALID CODE  **********

27A6  POP THE STACK
27A8  RETURN WITH POINTER
27AC  DIE HORBLY TO BAD INSTRUC.
27AF  RETURN
27B0 **************** ERROR RETURN ***************************************

27B0 RETURN WITH POINTER
27B4 EXIT WITH ERROR CODE
27B7 RETURN

27B8 **************** RELOCATE ABSOLUTE ADDRESS *************************

27B8 GET PAGE NUMBER TO CHECK
27BA GET NUMBER OF RANGES (LESS ONE) (283D)
27BD IS IT PRIOR TO START OF THIS RANGE? (283E)
27C0 YES? >>27C9
27C2 NO, IS IT AFTER END OF RANGE? (2846)
27C5 NO? >>27CD
27C9 ---
27CA CHECK EACH RANGE >>27BD
27CC RETURN

27CD ---
27CE ADD FUDGE FACTOR TO ADDRESS (284E)
27D1 AND UPDATE IT
27D3 RETURN

27D4 ********** BUMP POINTER TO NEXT ADDR *******************************

27D4 ---
27D5 ADD LENGTH TO POINTER
27DC CHECK TO SEE IF WE ARE DONE
27E2 ---
27E6 RETURN

27E7 *********** COMPUTE INSTRUCTION LENGTH *****************************

27E7 A-REG CONTAINS OPCODE
27EB ISOLATE LAST TWO BITS FOR LATER
27ED USE LAST 6 BITS AS TABLE INDEX
27EF GET BYTE WITH 4 LENGTHS IN IT (27FC)
27F2 ---
27F3 USING TOP TWO BITS AS INDEX... >>27F9
27F5 SHIFT DOWN THE PROPER LENGTH
27F9 AND ISOLATE IT IN A-REG
27FB RETURN

27FC ************* 6582 OP LENGTH TABLE *******************************

    EACH BYTE CONTAINS FOUR 2 BIT LENGTHS

ProDOS Relocator -- V1.0.1 -- 1 JAN 84

ATTR DESCRIPTION/CONTENTS

27FC ---

283C **************** RELOCATION DATA *******************************

283C ---
283D RELOCATE
283D NUMBER OF RANGES -- V1.0.1 -- 1 JAN 84
283E START
2846 END OF DATA/CONTENTS
284E ADDITION
2856 NOT U
285A PAD T
28CD

28FF ***************** RELOCATION CODE (3,2,1)***********************

28FF ---
2900 COPY # OF RANGES
2905 THE LA OF RANGE PAGES
2909 $3C/3D PF RANGE PAGES +1
2912 $3E/3F VS FACTORS
291D $42/$43 ED
2923 MAIN 5 NEXT PAGE BOUNDARY
2924 COPY $ 3
2929 SLOT 3
292C IS AT ** SET UP RAMDRIVE IN AUXMEM **************

2931 BUMP D
2937 ADD DE
293C RETURN

293D ********** --- $2A08

293D NOT U --- $2B00

2969 MEM TO AUX MEM COPY
2A08, L0290 TO AUXMEM $200 <C311>

2A00 ************* 2A00

(REFRESH COUNT (BF31)
VISES TO VOLUME SEARCH TABLE

2A00 SAVE
2A04 FORCE
2A0C COPY # 293D-29FF NOT USED **************

2A14 FIRST
2A17 NO? >>28ED

****

** RAMDRIVE DEVICE DRIVER ***********************

1D0 TO $200 IN AUXILIARY MEMORY)*

1D8 SET STORE SETTING (C018)
RAM READ/ WRITE (C000)
PUT PARAMETERS
TIME IN OR FORMAT COMMAND? (038A)
2A4D

**** FORMAT RAMDRIVE ****************************
ProDos Relocator -- V1.0.1 -- 1 JAN 84  
NEXT OBJECT ADDR: 2A19

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>2A19</td>
<td>YES, SAVE BLOCK WANTED</td>
</tr>
<tr>
<td>2A1D</td>
<td>ZERO SECTORS 5E AND 5F (031)</td>
</tr>
<tr>
<td>2A22'</td>
<td>COPY VOLUME NAME (SF3:&quot;RAM&quot;) (03D0)</td>
</tr>
<tr>
<td>2A25</td>
<td>TO VOLUME DIRECTORY BLOCK (0E04)</td>
</tr>
<tr>
<td>2A28</td>
<td>SFF</td>
</tr>
<tr>
<td>2A2E</td>
<td>HEX SFF'S TO TABLE (03C0)</td>
</tr>
<tr>
<td>2A34</td>
<td>ZERO (03C0)</td>
</tr>
<tr>
<td>2A39</td>
<td>COPY ACCESS/TOTAL BLOCKS TO (03D4)</td>
</tr>
<tr>
<td>2A3C</td>
<td>VOLUME DIRECTORY BLOCK (0E22)</td>
</tr>
<tr>
<td>2A42</td>
<td>REFORMAT? (03BA)</td>
</tr>
<tr>
<td>2A45</td>
<td>YES &gt;&gt;2A48</td>
</tr>
<tr>
<td>2A47</td>
<td>NO, DONE FIRST TIME PROCESSING (03BA)</td>
</tr>
<tr>
<td>2A4A</td>
<td>RESTORE BLOCK NUMBER (03BF)</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>2A4D</td>
<td>BLOCK NUMBER * 2 = SECTOR NUMBER (03BF)</td>
</tr>
<tr>
<td>2A53</td>
<td>SECTOR BEYOND MAIN MEMORY?</td>
</tr>
<tr>
<td>2A55</td>
<td>YES &gt;&gt;2A51</td>
</tr>
<tr>
<td>2A57</td>
<td>NO, SECTOR 6? (VOLUME BIT MAP)</td>
</tr>
<tr>
<td>2A59</td>
<td>NO &gt;&gt;2A5E</td>
</tr>
<tr>
<td>2A5B</td>
<td>YES, DUMMY UP A PHONEY BIT MAP SECTOR &gt;&gt;038A</td>
</tr>
<tr>
<td>2A5E</td>
<td>ELSE, READ/WRITE MAIN MEMORY SECTOR &gt;&gt;0340</td>
</tr>
</tbody>
</table>

******* READ/WRITE IN LANG CARD ******

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>2A61</td>
<td>SAVE SECTOR NUMBER</td>
</tr>
<tr>
<td>2A62</td>
<td>FIND IT IN MEMORY &lt;02E3&gt;</td>
</tr>
<tr>
<td>2A65</td>
<td>REMEMBER READ/WRITE STATUS</td>
</tr>
<tr>
<td>2A66</td>
<td>WRITING? &gt;&gt;2A68</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>2A69</td>
<td>NO, SECTOR FOLLOWS 1/0 4K AREA?</td>
</tr>
<tr>
<td>2A6B</td>
<td>YES &gt;&gt;2A71</td>
</tr>
<tr>
<td>2A6D</td>
<td>NO, FORCE IT TO SOXXX</td>
</tr>
<tr>
<td>2A6F</td>
<td>AND USE 2ND BANK OF CARD &gt;&gt;2A77</td>
</tr>
<tr>
<td>2A71</td>
<td>ELSE, USE 1ST BANK OF CARD (C083)</td>
</tr>
<tr>
<td>2A74</td>
<td>AND WRITE ENABLE IT (C083)</td>
</tr>
<tr>
<td>2A77</td>
<td>SAVE SECTOR NUMBER IN BLOCK (03BF)</td>
</tr>
<tr>
<td>2A7A</td>
<td>PRESERVE HIS BUFFER ADDR (03BD)</td>
</tr>
<tr>
<td>2A7E</td>
<td>ACROSS THE FOLLOWING: (03BD)</td>
</tr>
<tr>
<td>2A81</td>
<td>SELECT ALTERNATE ZEROPAGE (C089)</td>
</tr>
<tr>
<td>2A86</td>
<td>USE 5C00 AS A CROSS BANK XFER AREA (03BE)</td>
</tr>
<tr>
<td>2A8B</td>
<td>PRETEND THAT WAS CALLER'S BUFFER (03BD)</td>
</tr>
<tr>
<td>2A8E</td>
<td>AND SET UP POINTERS AGAIN &lt;02E3&gt;</td>
</tr>
<tr>
<td>2A92</td>
<td>COPY SECTOR TO 5C00</td>
</tr>
<tr>
<td>2A9D</td>
<td>THEN BACK TO MAIN ZERO PAGE (C080)</td>
</tr>
<tr>
<td>2AA8</td>
<td>READING OR WRITING?</td>
</tr>
<tr>
<td>2AA8</td>
<td>IF WRITING, DONE &gt;&gt;2A83</td>
</tr>
</tbody>
</table>

ProDos Relocator -- V1.0.1 -- 1 JAN 84  
NEXT OBJECT ADDR: 2AAA

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>2AAA</td>
<td>IF READING, ENABLE I.C. 1ST BANK (C0BB)</td>
</tr>
<tr>
<td>2AB0</td>
<td>AND COPY BLOCK SC00 TO HIS BUFFER &lt;02BC&gt;</td>
</tr>
<tr>
<td>2AB3</td>
<td>THEN EXIT &gt;&gt;03DC</td>
</tr>
<tr>
<td>2AB6</td>
<td>IF WRITING, COPY HIS BLOCK TO SC00 &lt;02BC&gt;</td>
</tr>
<tr>
<td>2AB9</td>
<td>THEN DO COMMON CODE ABOVE &gt;&gt;02E8</td>
</tr>
</tbody>
</table>

2ABC ************ COPY BLOCK IN MAIN 48K ************

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>2AB7</td>
<td>THIS ENTRY COPIES SECTOR $0C/$D</td>
</tr>
<tr>
<td>2ABE</td>
<td>THIS ENTRY COPIES ANY BLOCK (03BF)</td>
</tr>
<tr>
<td>2AC1</td>
<td>FIND SECTOR/SET POINTERS &lt;02E3&gt;</td>
</tr>
<tr>
<td>2AC4</td>
<td>WRITING? &gt;&gt;2AD9</td>
</tr>
<tr>
<td>2ACA</td>
<td>NO, WRITE TO MAIN 48K RAM (C084)</td>
</tr>
<tr>
<td>2AD0</td>
<td>COPY BLOCK AUX MEM --&gt; MAIN MEM</td>
</tr>
<tr>
<td>2AD5</td>
<td>WRITE TO AUX MEM AGAIN (C005)</td>
</tr>
<tr>
<td>2AD8</td>
<td>DONE</td>
</tr>
<tr>
<td>2AD9</td>
<td>---</td>
</tr>
<tr>
<td>2ADB</td>
<td>GO BACK TO MAIN MEM LANG CARD.. (03ED)</td>
</tr>
<tr>
<td>2ADE</td>
<td>TO COPY MAIN MEM --&gt; AUX MEM</td>
</tr>
</tbody>
</table>

2AE3 ************ FIND RAM SECTOR/SET POINTERS ************

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>2AE3</td>
<td>GET COMMAND (03BB)</td>
</tr>
<tr>
<td>2AE6</td>
<td>READ OR WRITE?</td>
</tr>
<tr>
<td>2AE7</td>
<td>WRITE? &gt;&gt;2B06</td>
</tr>
<tr>
<td>2AE9</td>
<td>NO, READ OR FORMAT (03BE)</td>
</tr>
<tr>
<td>2AF0</td>
<td>542/43 --&gt; BUFFER IN HIS MEMORY (03BD)</td>
</tr>
<tr>
<td>2AF3</td>
<td>540/41 --&gt; SECOND PAGE OF SAME</td>
</tr>
<tr>
<td>2AE7</td>
<td>GET PAGE NUMBER (03BF)</td>
</tr>
<tr>
<td>2AF0</td>
<td>33C/3D --&gt; BLOCK IN /RAM DRIVE</td>
</tr>
<tr>
<td>2AF0</td>
<td>33E/3F --&gt; SECOND PAGE OF SAME</td>
</tr>
<tr>
<td>2BF0</td>
<td>ALWAYS BRANCH AROUND WRITE CODE &gt;&gt;2B21</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>2B06</td>
<td>WRITE, (03BE)</td>
</tr>
<tr>
<td>2B0D</td>
<td>33C/3D --&gt; BUFFER IN HIS MEMORY (03BD)</td>
</tr>
<tr>
<td>2B10</td>
<td>35E/3F --&gt; SECOND PAGE OF SAME</td>
</tr>
<tr>
<td>2B17</td>
<td>542/43 --&gt; BLOCK IN /RAM DRIVE</td>
</tr>
<tr>
<td>2B19</td>
<td>540/41 --&gt; SECOND PAGE OF SAME</td>
</tr>
<tr>
<td>2B21</td>
<td>SECOND PAGE EXISTS FIRST</td>
</tr>
<tr>
<td>2B25</td>
<td>EXIT</td>
</tr>
</tbody>
</table>

2B26 ************ RETURN WITH DUMMY SECTOR ************
Beneath Apple ProDOS Supplement

2B26 ZERO SECTOR SC/D AND SELECT IT <032F>
2B29 COPY TO/FROM HIS BUFFER <02C1>
2B2C AND EXIT >> 03DC

2B2F ************ ZERO BLOCK BUFFER ********************************************

032F
2B2F ZERO SECTOR SC/D ENTRY
0331
2B31 ZERO ANY GIVEN SECTOR ENTRY (03BF)
0334
2B34 FIND SECTOR/SET POINTERS <02E3>
2B3B ZERO BOTH PAGES OF BLOCK
2B3F AND EXIT

2B40 ********** READ/WRITE IN LOW 48K ***************************************

0340
2B40 SECTOR 4 (VOLUME DIRECTORY)?
2B42 NO >> 2B48
2B44 YES, MAKE THAT BLOCK 7 INSTEAD
2B46 AND GO DO I/O NOW >> 2B56
2B48 ELSE, LESS THAN SECTOR SD?
2B4A IF SO, PASS BACK SC ZEREOED >> 2B26
2B4C START MSB AT ZERO
2B4E GET ORIGINAL BLOCK NUMBER
2B50 BLOCK $5D THROUGH $5F?
2B52 NO >> 2B59
2B54 YES, ADJUST TO $D THROUGH $F
2B56 AND USE $A000 THRU $A0FF IN /RAM >> 0383
2B59 ELSE, FOR SECTORS $D THRUS $5C
2B5A SUBTRACT 8
2B5C AND DIVIDE BY 17 ($11)
2B62 XRQ IS QUOTIENT
2B66 AND AREQ IS REMAINDER
2B67 REMAINDER OF 1?
2B69 NO >> 2B71
2B6B YES, EVERY 17TH BLOCK GOES...
2B6C AT $1200, $1400, $1600, $1800
2B6D BY ADDING 8
2B6F AND GO DO IT >> 2B83
2B71 BUMP QUOTIENT (START AT $2XXX)
2B73 SHIFT IT TO TOP NIBBLE OF BYTE
2B77 GOT A REMAINDER? >> 2B7F
2B7D IF SO, DECREMENT IT (NOT USING 1)
2B7F THEN ADD INTO TOP NIBBLE
2B80 TO FORM $14 THRU $4F (03BF)
0383
2B83 BLOCK*2 FOR SECTOR NUMBER
2B84 COPY THE BLOCK <02BE>

ProDOS Relocator -- V1.0.1 -- 1 JAN 84 NEXT OBJECT ADDR: 2B26

ADDRES DESCRIPTION/CONTENTS

2B26 ZERO SECTOR SC/D AND SELECT IT <032F>
2B29 COPY TO/FROM HIS BUFFER <02C1>
2B2C AND EXIT >> 03DC

2B2F ************ ZERO BLOCK BUFFER ********************************************

032F
2B2F ZERO SECTOR SC/D ENTRY
0331
2B31 ZERO ANY GIVEN SECTOR ENTRY (03BF)
0334
2B34 FIND SECTOR/SET POINTERS <02E3>
2B3B ZERO BOTH PAGES OF BLOCK
2B3F AND EXIT

2B40 ********** READ/WRITE IN LOW 48K ***************************************

0340
2B40 SECTOR 4 (VOLUME DIRECTORY)?
2B42 NO >> 2B48
2B44 YES, MAKE THAT BLOCK 7 INSTEAD
2B46 AND GO DO I/O NOW >> 2B56
2B48 ELSE, LESS THAN SECTOR SD?
2B4A IF SO, PASS BACK SC ZEREOED >> 2B26
2B4C START MSB AT ZERO
2B4E GET ORIGINAL BLOCK NUMBER
2B50 BLOCK $5D THROUGH $5F?
2B52 NO >> 2B59
2B54 YES, ADJUST TO $D THROUGH $F
2B56 AND USE $A000 THRU $A0FF IN /RAM >> 0383
2B59 ELSE, FOR SECTORS $D THRUS $5C
2B5A SUBTRACT 8
2B5C AND DIVIDE BY 17 ($11)
2B62 XRQ IS QUOTIENT
2B66 AND AREQ IS REMAINDER
2B67 REMAINDER OF 1?
2B69 NO >> 2B71
2B6B YES, EVERY 17TH BLOCK GOES...
2B6C AT $1200, $1400, $1600, $1800
2B6D BY ADDING 8
2B6F AND GO DO IT >> 2B83
2B71 BUMP QUOTIENT (START AT $2XXX)
2B73 SHIFT IT TO TOP NIBBLE OF BYTE
2B77 GOT A REMAINDER? >> 2B7F
2B7D IF SO, DECREMENT IT (NOT USING 1)
2B7F THEN ADD INTO TOP NIBBLE
2B80 TO FORM $14 THRU $4F (03BF)
0383
2B83 BLOCK*2 FOR SECTOR NUMBER
2B84 COPY THE BLOCK <02BE>

ProDOS Relocator -- V1.0.1 -- 1 JAN 84 NEXT OBJECT ADDR: 2B87

ADDRES DESCRIPTION/CONTENTS

2B87 THEN EXIT >> 03DC

2B8A ********** READ/WRITE BIT MAP BLOCK **********************************

038A
2B8A USE SC/D AS A DUMMY SECTOR
2B8F GO FIND IT AND SET POINTERS <02E3>
2B92 WRITING? >> 2BA7
2B94 NO, READING - ZERO BLOCK AT SC/D <0334>
2B99 COPY BIT MAP IMAGE TO DUMMY BLOCK (03C0)
2BA1 COPY BLOCK BACK TO CALLER'S BUFFER <02C1>
2BA4 THEN EXIT >> 03DC
2BA7 WRITING, COPY CALLER'S BUFFER TO SC/D <02E3>
2BAA FIND SC/D AND SET POINTERS <02E3>
2BAF COPY FROM SECTOR TO BIT MAP IMAGE
2BB7 THEN EXIT >> 03DC

2BBA ********** RAM DRIVE DATA (AT $3BA) *******************************

039A
2BBA FIRST TIME ENTRY FLAG
03BB COMMAND FROM PARM LIST
03BC UNIT NUMBER FROM PARM LIST
03BD
03BE
2BB0 BUFFER ADDRESS FROM PARM LIST
03BF
2BBF BLOCK NUMBER FROM PARM LIST
03C0
2BC0 BIT MAP IMAGE FOR RAM DRIVE

03D0
2BD0 /RAM VOLUME NAME
2BD1 'RAM'
03D4
2BD4 ACCESS, ENTRY LENGTH
2BD6 NUMBER OF ENTRIES
2BD7 FILE COUNT
2BD9 BIT MAP BLOCK POINTER
2BDB BLOCKS ON DISK
ProDOS Relocator -- V1.0.1 -- 1 JAN 84

NEXT OBJECT ADDR: 2BDC

---

ADDR DESCRIPTION/CONTENTS
---

2BDC ********** EXIT TO MAIN MEMORY ***********************

03DC ,
  2BDC WRITE ENABLE RAM CARD (C08B)
  2B3F RESTORE $STORE STATUS $2BEA
  2BE5 $STORE WAS ON (C001)
  2BEA GO AROUND PARM TO XFER >>03EF
03ED CROSS BANK XFER ADDRESS LSB
  03EE AND MSB
03EF 2BFE RETURN TO $FF44 (NORMAL EXIT)
  2BF8 USE ROM XFER ROUTINE TO DO IT >>C314

2C88 ********** DISK DEVICE DRIVER FOR /RAM *******************
           (COPIED TO $FF80 IN KERNEL)

2C88 --
2C89 SAVE SPAGE STUFF I WILL CLOBBER
2C85 FROM $3C THRU $47 (FF81)
2C80 SAVE $3ED/E (CROSS BANK XFER ADDR) (03ED)
2C8F COMMAND = STATUS?
2C92 IF SO, SIMPLE EXIT WILL DO >>2C44
2C90 ELSE, TOO BIG A COMMAND NUM?
2C91 IF SO, ERROR >>2C3B
2C92 ELSE, INVERT BITS OF CMD
2C90 AND SAVE IT
2C92 FORMAT? >>2C2C
2C94 NO, CHECK BLOCK NUMBER
2C98 MUST BE <128 FOR /RAM
2C9C GOING TO $200 IN AUX MEMORY
FF33 USE XFER TO GET THERE >>C314

2C38 I/O ERROR RETURN CODE
2C3D EXIT >>2C43
2C3F WRITE PROTECTED RETURN CODE
2C41 --
2C42 ERROR EXIT >>2C47
2C44 NORMAL EXIT, RETURN CODE IS 0
2C47 ---
2C4B RESTORE ZERO PAGE IS USED (FF81)
2C53 AND $3ED/E (FF7F)
2C61 AND EXIT TO CALLER WHEN THRU

ProDOS Relocator -- V1.0.1 -- 1 JAN 84

NEXT OBJECT ADDR: 2C61

---

ADDR DESCRIPTION/CONTENTS
---

2C62 ********** COPY MAIN TO AUX BLOCK *********************
           (CALLED FROM AUX MEM HANDLER)

FF62
  2C62 WRITE IN AUX 40K (C805)
  2C67 COPY BOTH PAGES OF BLOCK
  2C72 WRITE IN MAIN 40K AGAIN (C804)
  2C77 GO TO $2D0 IN AUX MEMORY TO RETURN (03ED)
  2C7C RETURN TO AUX MEM HANDLER AGAIN >>FF33

2C7F ********** DATA AREA ***********************

FF7F
  2C7F SAVED XFER ADDRESS
FF80

FF81
  2C81 ZERO PAGE SAVE AREA

2C8D ********** NOT USED ***********************

2C8D --

2D00 ********** START OF PRODOS LOAD IMAGE *****************

2D00 LOAD IMAGE AT $2D00

2D00 --
### Beneath Apple ProDOS Supplement

<table>
<thead>
<tr>
<th>Address</th>
<th>Description/Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>D000--D07F</td>
<td>Module Starting Address</td>
</tr>
<tr>
<td>D080--D0FF</td>
<td>Pointer to callers pmlist</td>
</tr>
<tr>
<td>D040</td>
<td>Bad call type</td>
</tr>
<tr>
<td>D001</td>
<td>Bad parameter count</td>
</tr>
<tr>
<td>D004</td>
<td>Bad parameter count</td>
</tr>
<tr>
<td>D025</td>
<td>Interrupt Table full</td>
</tr>
<tr>
<td>D027</td>
<td>I/O Error</td>
</tr>
<tr>
<td>D028</td>
<td>No device connected</td>
</tr>
<tr>
<td>D02A</td>
<td>Write protected</td>
</tr>
<tr>
<td>D02E</td>
<td>Volume switched</td>
</tr>
<tr>
<td>D048</td>
<td>I/O Pointer - Index Block or..</td>
</tr>
<tr>
<td>D049</td>
<td>caller's pathname buffer pointer</td>
</tr>
<tr>
<td>D04A</td>
<td>I/O Pointer - Data Block</td>
</tr>
<tr>
<td>D04B</td>
<td>I/O Pointer - Data Block</td>
</tr>
<tr>
<td>D04C</td>
<td>I/O Pointer - Caller's Data or..</td>
</tr>
<tr>
<td>D04D</td>
<td>buffer pointer passed in pmlist or..</td>
</tr>
<tr>
<td>D04E</td>
<td>old I/O buffer</td>
</tr>
<tr>
<td>D048</td>
<td>I/O Pointer - Index Block or..</td>
</tr>
<tr>
<td>D049</td>
<td>caller's pathname buffer pointer</td>
</tr>
<tr>
<td>D04A</td>
<td>I/O Pointer - Data Block</td>
</tr>
<tr>
<td>D04B</td>
<td>I/O Pointer - Data Block</td>
</tr>
<tr>
<td>D04C</td>
<td>I/O Pointer - Caller's Data or..</td>
</tr>
<tr>
<td>D04D</td>
<td>buffer pointer passed in pmlist or..</td>
</tr>
<tr>
<td>D04E</td>
<td>old I/O buffer</td>
</tr>
<tr>
<td>D080</td>
<td>Jump to MLI entry</td>
</tr>
<tr>
<td>D093</td>
<td>JSPARE (Jump to SPARE)</td>
</tr>
<tr>
<td>D0A6</td>
<td>DATETIME vector</td>
</tr>
<tr>
<td>D0A9</td>
<td>Jump to System Error handler</td>
</tr>
<tr>
<td>D0AF</td>
<td>Jump to System Dec</td>
</tr>
</tbody>
</table>
1 JAN 84 NEXT OBJECT ADDR: D000

INTENTS

Table 1

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>D04F</td>
<td>no</td>
</tr>
<tr>
<td>D050</td>
<td>$8X - Calls to I/O Drivers &gt;&gt;D066</td>
</tr>
<tr>
<td>D052</td>
<td>$CX/$DX - Non System calls &gt;&gt;D071</td>
</tr>
<tr>
<td>D054</td>
<td>Else, $4X - Interrupt support</td>
</tr>
<tr>
<td>D055</td>
<td>Isolate type (DEALLOC = 1, ALLOC = 0)</td>
</tr>
<tr>
<td>D057</td>
<td>Call Interrupt Support &lt;D0F3&gt;</td>
</tr>
<tr>
<td>D05A</td>
<td>Then Exit to Caller &gt;&gt;D078</td>
</tr>
<tr>
<td>D05D</td>
<td>Go to quit code via global page &gt;&gt;B0F3</td>
</tr>
<tr>
<td>D060</td>
<td>***************************************************************</td>
</tr>
<tr>
<td></td>
<td>**** MLI GET_TIME CALL ****</td>
</tr>
<tr>
<td></td>
<td>********************************</td>
</tr>
<tr>
<td>D066</td>
<td>Call Date/Time driver &lt;B0F6&gt;</td>
</tr>
<tr>
<td>D063</td>
<td>and exit to caller &gt;&gt;D078</td>
</tr>
<tr>
<td>D066</td>
<td>***************************************************************</td>
</tr>
<tr>
<td></td>
<td>**** MLI READ_BLOCK CALL ****</td>
</tr>
<tr>
<td></td>
<td>**** MLI WRITE_BLOCK CALL ****</td>
</tr>
<tr>
<td></td>
<td>********************************</td>
</tr>
<tr>
<td></td>
<td>$00 - Read Block</td>
</tr>
<tr>
<td></td>
<td>$81 - Write Block</td>
</tr>
<tr>
<td>D066</td>
<td>***</td>
</tr>
<tr>
<td>D067</td>
<td>Set $42 -&gt; 1 for READ, 2 for WRITE</td>
</tr>
<tr>
<td>D06B</td>
<td>Do Block I/O &lt;D0B2&gt;</td>
</tr>
<tr>
<td>D06E</td>
<td>Then Exit to Caller &gt;&gt;D078</td>
</tr>
<tr>
<td>D071</td>
<td>***************************************************************</td>
</tr>
<tr>
<td></td>
<td>$CX and $DX CALLS ***************************************************************</td>
</tr>
<tr>
<td>D071</td>
<td>***</td>
</tr>
<tr>
<td>D072</td>
<td>Isolate function Index</td>
</tr>
<tr>
<td>D075</td>
<td>Perform function and exit to caller &lt;D23C&gt;</td>
</tr>
<tr>
<td>D078</td>
<td>***************************************************************</td>
</tr>
<tr>
<td></td>
<td>EXIT TO CALLER ***************************************************************</td>
</tr>
<tr>
<td>D078</td>
<td>Clear Backup</td>
</tr>
<tr>
<td>D080</td>
<td>Error occurred?</td>
</tr>
<tr>
<td>D083</td>
<td>Save test results</td>
</tr>
<tr>
<td>D084</td>
<td>Disable interrupts</td>
</tr>
<tr>
<td>D085</td>
<td>MLI no longer active (BF9B)</td>
</tr>
<tr>
<td>D088</td>
<td>Get test results back</td>
</tr>
<tr>
<td>D089</td>
<td>Store in X reg</td>
</tr>
<tr>
<td>D08A</td>
<td>Set up Return Address on stack (BF9D)</td>
</tr>
<tr>
<td>D092</td>
<td>Put test results on stack</td>
</tr>
<tr>
<td>D094</td>
<td>Put error code in A reg</td>
</tr>
<tr>
<td>D095</td>
<td>Restore X reg (BF9E)</td>
</tr>
<tr>
<td>D098</td>
<td>Restore Y reg (BF9F)</td>
</tr>
<tr>
<td>D09B</td>
<td>Put error code on stack</td>
</tr>
<tr>
<td>D09C</td>
<td>Get RAM/RAM orientation (BF4)</td>
</tr>
<tr>
<td>D09F</td>
<td>Exit via RAM Global Page &gt;&gt;BFA0</td>
</tr>
</tbody>
</table>
Beneath Apple ProDOS Supplement

---

ProDOS MLI -- V1.0.1 -- 1 JAN 84

ADDRESS DESCRIPTION/CONTENTS

---

D0A2 *********** NO DEVICE CONNECTED ***********

D0A2 ---
D0A4 Call System Error Handler (Global Page) <BF09>

D0A7 ********** BAD SYSTEM CALL NUMBER ***************

D0A7 ---
D0A9 Branch always taken >>0AD

D0AB ********** BAD PARAMETER COUNT ***********************

D0AB ---
D0AD Call System Error Handler <DFD7>
D0B0 Exit to Caller >>D78

D0B2 ********** BLOCK I/O SETUP ***********************

D0B2 ---
D0B4 Save Old Processor Flags
D0B5 Disable Interrupts
D0B6 Copy Parameters to $43-$47
D0B8 Save Starting Buffer Page in $4F
D0C3 Find last page + 1
D0C6 Round up if Buffer not page aligned >>D0C9
D0C9 Is this Memory already in use? <EEB9>
D0CC No, then exit with error >>D0D6
D0CE No, do Block I/O <D0DA>
D0D1 Error? >>D0D6
D0D3 No, then exit normally
D0D5 RETURN
D0D6 Error Exit
D0D7 Call System Error Handler <BF09>

D0DA ********** Block I/O ***********************

D0DA ---
D0DC Force off unused UNIT bits
D0E3 Put Drive number in X reg
D0F7 Put Device Handler Address in Jump Vector (F0B5)
D0F0 Exit through Device Handler >>F0B5

D0F3 ********** Interrupt Handler ***********************

ALLOC/DEALLOC

D0F3 Save Call Type
D0F5 Which Type?
D0F6 DEALLOC? >>124

---

D14E And RTI Address (BF8E)
D155 Replace stack to original condition
D159 Save active slot index (D1C4)
D15C Is stack full?
D15F Pop off 16 bytes and save them
D161 ---
D168 Save SPA - SFF (top of zero page)
D16A ---
D172 Is there a User Vector #1 (BF81)
D175 No >>D17C
D177 Yes, call it <<D1CE
D17A His interrupt? >>D19F
D17C Is there a User Vector #2 (BF83)
D17F No >>D186
D181 Yes, call it <<D1D1
D184 His interrupt? >>D19F
D186 Is there a User Vector #3 (BF85)
D189 No >>D190
D18B Yes, call it <<D1D4

---

ProDOS MLI -- V1.0.1 -- 1 JAN 84

ADDRESS DESCRIPTION/CONTENTS

---

ALLOC

D0F8 ---
D0FA Look for empty slot (BF7E)
D101 His Address better be non-zero
D105 Store Address of His routine in Global Page (BF7E)
D10E And return the position number we used
D114 Exit
D115 Skip this Vector
D117 Last one?
D119 No, check another >>D0FA
D11B Yes, Table Full Error
D11D Always taken >>D121
D11F Bad Parameter Error
D121 Call System Error Handler <BF09>

DEALLOC

D124 ---
D126 Get Position Number
D128 Can't be zero >>D11F
D12C Or greater than 4 >>D11F
D12F Make Index into Table from it
D132 And zero His Vector (BF7E)
D139 Then Exit

D13A ********** IRQ Handler ***********************

D13A ---
D13C Save A reg from Monitor (BF88)
D13F And X,Y,Z and P (BF89)
D1F6  Blank out a line
D1F8  ---
D1FD  Print "INSERT SYSTEM DISK AND RESTART" (BFDS)
D207  Go into infinite loop if no error code >>D239
D20B  "-" (O7F1)
D210  "s" (O7F2)
D215  "r" (O7F3)
D21A  "p" (O7F4)
D21C  Convert error code to Hex
D228  And print it (O7F6)
D22C  Second digit also
D239  Infinite loop >>D239

D23C  ********** PERFORM FILING OR ***************
       HOUSEKEEPING FUNCTIONS ********

D23C  Save function index (F077)
D23F  Get INFO flags for this command (EF8D)
D242  Times 2
D243  Store Command Number times 2 (F073)
D248  And use it to index into Address Table
D24C  Set up Jump Vector with this funtion's (F075)

D29F  Length + 1 (F08D)
D2A3  Get first character of
D2A7  Is it "/"?
D2A9  No >>D2AF
D2AB  Yes - indicate fully
D2AE  Bump past "/"
D2AF  ---
D2B1  Length of Index Level
D2B4  First character of Index
D2B7  Start of index info
D2BA  At end of name yet?
D2BD  Yes >>D2F4
D2BE  No - get next character
D2C5  Is it "/"?
D2C7  Yes >>D30F
D2C9  No - lower case
D2CA  No >>D2CF
D2CB  Validility check <<D7F
D2CD  Yes - force upper case
D2CF  Copy to my Pathname buff? (F073)
D2D2  Increment index level
D2D5  Subsequent characters (C5)
D2D7  Increment index level

Pathname

Beneath Apple ProDOS Supplement

ProDOS MLI -- V1.0.1 -- 1 JAN 84

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>D18E</td>
<td>His interrupt? &gt;&gt;D19F</td>
</tr>
<tr>
<td>D190</td>
<td>Is there a User Vector #4 (BF87)</td>
</tr>
<tr>
<td>D193</td>
<td>No &gt;&gt;D19A</td>
</tr>
<tr>
<td>D195</td>
<td>Yes, call it &lt;&lt;D17</td>
</tr>
<tr>
<td>D19B</td>
<td>His interrupt? &gt;&gt;D19F</td>
</tr>
<tr>
<td>D19C</td>
<td>Indicate error type 1</td>
</tr>
<tr>
<td>D19D</td>
<td>Call System Death Handler &lt;&lt;BF0C</td>
</tr>
<tr>
<td>D19F</td>
<td>Interrupt Service</td>
</tr>
<tr>
<td>D1A0</td>
<td>Restore zero page (EFA5)</td>
</tr>
<tr>
<td>D1A9</td>
<td>And stack (BF8B)</td>
</tr>
<tr>
<td>D1B9</td>
<td>Reload X and Y (BF8A)</td>
</tr>
<tr>
<td>D1BF</td>
<td>Disable I/O ROMS (CFFF)</td>
</tr>
<tr>
<td>D1C2</td>
<td>Replace active slot number (C100)</td>
</tr>
<tr>
<td>D1CB</td>
<td>Exit from Interrupt &gt;&gt;BF8D</td>
</tr>
<tr>
<td>D1CE</td>
<td>User Interrupt Handlers (#1 - #4) &gt;&gt;BF80</td>
</tr>
</tbody>
</table>

D1DA  ********** SYSTEM ERROR HANDLER ***************

D1DA  Save Error Code (BF0F)
D1DE  Pop out of subroutine
D1DF  Exit to caller with Error Code (BF0F)
D1E3  RETURN

D1E4  ********** SYSTEM DEATH HANDLER ***************

D1E4  ---
D1E6  ; Entry from System Global Page here
D1E7  Turn off 80 column card (C00C)
D1E8  ; Check length of caller
D1E9  Check length of caller (C561)
D1EC  ; Entry from System Global Page here
D1EF  ; Check length of caller
D1F1  ; Check length of caller (C561)

D24F  Length + 1 (F08D)
D253  Get first character of
D257  Is it "/"?
D259  No >>D5AF
D25B  Yes - indicate fully
D25C  Bump past "/"
D25D  ---
D261  Length of Index Level
D264  First character of Index
D267  Start of index info
D26A  At end of name yet?
D26D  Yes >>D6F4
D26F  No - get next character
D275  Is it "/"?
D277  Yes >>D30F
D279  No - lower case
D27B  No >>D2CF
D27C  Validility check <<D7F
D27D  Yes - force upper case
D27F  Copy to my Pathname buff? (F073)
D282  Increment index level
D285  Subsequent characters (C5)
D287  Increment index level

Pathname
D2A First character must be alphabetic >>D2E8
D2C Is it "."?
D2E Yes - get next character >>D2BA
D280 No - is it special or control character
D2E2 Yes - Bad Pathname then >>D2F0
D2E4 Is it numeric?
D2E6 Yes - get next character >>D2BA
D2E8 Is it Alphabetic?
D2EE If so get next character >>D2BA
D2F0 Else
D2F1 Bad Pathname
D2F3 RETURN
D2F4 ---
D2F6 Any characters in last Index level? (F078)
D2F9 Yes >>D2FF
D2FB No, zero characters in it (F078)
D2FF And toss out last "/"
D2F ---
D300 Mark end of name with $00 (F100)
D303 Name too long? >>D2F0
D305 No - save final length (F05E)
D308 Set X - & 0
D30C Last Index more than 15 characters?
D30E Yes - then no good >>D2F0
D310 Save output index (F07D)
D313 Store length of previous Index level (F07A)
D316 Just before it in buffer (F100)
D319 Restore output index (F07D)
D31C And continue >>D2AF
D31E End of Name
D31F Fully qualified name? (F07C)
D322 Yes >>D329
D324 No - Got a Prefix (BF9A)
D327 No - error >>D2F0
D329 Else, okay to exit

D32A *****************************************************
****** MLI SET_PREFIX CALL ******
*****************************************************

D32A Copy Pathname <D27F>
D32D It's okay >>D339
D32F Check length of Volume name (F100)
D334 If zero - no Prefix wanted (BF9A)
D337 Exit with no error
D338 RETURN

D339 Get File entry for last index <D79B>
D33C Okay? >>D342
D33E Invalid Pathname?
D340 No - Out now! >>D380
D342 Sub Directory file? (F01F)
D349 No, error >>D37E
D34B Fully qualified path? (F07C)
D34E Yes >>D353
D350 No - use old Prefix also (BF9A)
D353 ---
D355 Compute new Prefix Index (F05E) (F100)
D358 Does new Prefix exceed 64 characters?
D35A Yes - Bad Path error >>D2F0
D35D Store new Prefix pointer (BF9A)
D363 Set Device Number of Prefix Directory (F3)
D369 Save Keyblock for Prefix Directory (F068)
D372 Copy Prefix to top of Path buffer (F100)
D375 (preceded by old Prefix if one exists)
D37D Exit normally
D37E Bad File Type Error
D380 ---
D381 RETURN

D382 *****************************************************
****** MLI GET_PREFIX CALL ******
*****************************************************

D382 Set (S4E) -> Data Buffer
D38E Set Length = 64 (max)
D394 Validity check buffer storage <EE6C>
D39B Error? >>D380
D39F Get Prefix index (BF9A)
D3A3 No Prefix? - Length = 0 >>D3A9
D3A5 Compliment for length
D3A9 Store in first byte of buffer
D3AB If null Prefix exit >>D3C3
D3AD ---
D3AE Copy Prefix to caller's buffer replacing:
D3BF index level name length bytes with "/
D3BB ---
D3BF End it with a "/
D3C3 ---
D3C4 Exit normally
ProDOS MLI -- V1.0.1 -- 1 JAN 84
NEXT OBJECT ADDR: D3C5

D3C5 *********** VALIDITY CHECK REFERENCE NUMBER **************
          (PASSED BY CALLER)

D3C5 Get Reference Number
D3C9 If zero then no good >>D426
D3CD If > 8 then no good >>D426
D3CF Save Reference Number
D3D8 Multiply by 32
D3D6 Result gives offset into FCB's (F052)
D3DA Get back Reference Number
D3DB File Control Block active this Reference? (F00)
D3DE No - Bad Reference Number >>D421
D3D6 Get Buffer Number (F08B)
D3D3 Find Buffer address in Global Page <E26>
D3D9 No Buffer? >>D412
D3DB Buffer okay, save Page Pointer in $48
D3DF Second block in $49
D3F1 Set last Device used in Global Page (F081)
D3F7 Finish setting up pointers (F08D)
D3FA (S4A) -> 1st Block of Buffer (data)
D3FC (S4B) -> 2nd Block of Buffer (index)
D3FE ---
D3FF Search all Volume Control Blocks (F010)
D402 For the one which goes with requested unit (F081)
D407 ---
D40D Can't find matching Volume Control Block
D40F So die with error type 38A <BF0C>
D412 No Buffer in open File Control Block
D414 So die with error type 38B <BF0C>
D417 Is Volume mounted? (F080)
D41A No, keep looking >>D407
D41C Save Volume Control Block index (F051)
D420 Exit normally
D421 ---
D423 !!!!!! (F052)
D426 Bad Reference Number error
D429 RETURN

D42A ********** MLI ONLINE CALL **********

D42A Set ($48) -> Data Buffer <E03>
D42D Set Length = 0
D437 Get Unit Number
D439 Do all Units? >>D442
D43B No, just one
D43D Set length = 16 (F09A)
D440 Always taken >>D447
D442 If all Units

ProDOS MLI -- V1.0.1 -- 1 JAN 84
NEXT OBJECT ADDR: D444

D444 Set Length = 256 (maximum) (F09B)
D447 Is Buffer in main RAM? <EE6C>
D44A No, then exit >>D47F
D44C Yes, zero out Buffer
D451 ---
D456 Index into Data Buffer = 80 (F07A)
D45B Get Unit Number again
D45D Isolate valid bits
D45F Specific Unit requested? >>D480
D461 No, copy Device List from Global Page <D57>
D464 Save Device Count (F07D)
D467 Get last Device (F08A)
D46A Generate return data for it <D480>
D46D Bump data buffer index by 16 (F07A)
D46E Get next Device (F07D)
D47A And go do it >>D464
D47C When done, exit
D47F RETURN

D480 Save Device Number (F030)
D483 Scan for the Volume Control Block <D0A9>
D486 Error? >>D4B8
D488 We need Block 2 (Key Block of VolDir)
D490 Read Volume Directory Key Block <D061>
D493 Error? >>D4B8
D495 Was something already mounted? (F051)
D49B No >>D4A2
D49D No, is this open? (F011)
D4A0 Yes >>D4AE
D4A2 No, set up Volume Control Block for new VOL <D0C4>
D4A5 Error? >>D4B8
D4A7 No
D4A9 Was a duplicate Volume Control Block found? (F075)
D4AC Yes, then error >>D4B8
D4AE See if the same volume is still there (F051)
D4B4 If not, Disk Switch Error
D4B6 Else, all is well - continue >>D4C6

D4B8 ******** ERROR *****************

        Store code in data buffer entry
D4B8 ---
D4B9 Store Device Number in entry <D4E8>
D4BE Store error code next
D4C0 Duplicate Volume error?
D4C2 No - done >>D4D4
D4C5 Store Device Number for duplicate next (F076)
D4CD No Duplicate now
D4D4 Exit with error
D4D5 RETURN
D4D6 *********** MAKE ONLINE VOLUME ENTRY *******************************
D4D6 Get name length for loop index (F200)
D4DF Copy name to Buffer entry (F200)
D4E6 Done yet? (F078)
D4E9 No, do another >>D4DF
D4EB Yes, find current Buffer entry (F07A)
D4EE Store Device number (BF30)
D4F6 Return to caller

D4F7 **********************************************
***** MLI CREATE CALL *****
**********************************************
D4F7 Follow Path to File <D7AB>
D4FA Error? - I'm expecting one >>D500
D4FC If File was found - Duplicate error
D4FE --
D4FF Return to caller

D500 File not found?
D502 No, then a real error occured >>D4FE
D504 Yes, get requested storage type
D506 Is it $0, $01, $02 or $03?
D50A Yes, carry on >>D518
D50C Is it $0D?
D50E No, then exit with error >>D520
D510 Get status of this device (BF30)
D516 Exit on error >>D523
D518 Is there a free Directory entry? (F058)
D51B No >>D524
D51D Yes - continue >>D586
D520 Indicate Bad Storage Type
D523 Return to caller

D524 Is this the Volume Directory? (F006)
D52A No, we can extend it >>D530
D52C Yes, indicate Volume Directory Full error
D52F Return to caller

* EXTEND DIRECTORY FILE *

D530 Save old current Block number
D536 Allocate a Block on Disk <DCA9>
D539 Save the number
D53A Replace BLKNUM
D540 Was there a free Block?
D541 No, then exit >>D523
D543 Yes, set up forward pointer in old one (F002)
ProDOS MLI -- V1.0.1 -- 1 JAN 84

ADDR DESCRIPTION/CONTENTS

D5EE Copy File name to File Entry Buffer (F07A)
D5FC Copy caller's Access Byte
NOTE: This should be validity checked!!

D604 and copy File type
D609 ---
D60A and AUX TYPE
D613 Copy Version and Min Version (0,0) (EF80)
D616 constants to entry (F03B)
D61F Indicate 1 Block used
D624 Copy Directory Header Block number (F01A)
D633 Is this a directory file?
D635 Yes >>D6EE
D637 No, Directory file - Build Header in $F600
D639 Copy completed Directory entry (F01F)
D63C to $F600 buffer first (F064)
D640 Loop until done >>D639
D642 Make Storage type $E in Header itself
D647 Put "HUSTON" (Author) in Reserved area
D64F and Version, Min Version, Access, (EF80)
D652 Entry-length, File count and (F620)
D655 Parent pointer from constants
D656 Loop until done >>D649
D65A Copy Parent Block entry number (F01C)
D661 Loop until done >>D65A
D663 Copy Parent entry Length (F011)
D66B EOF = $200 (F035)
D66E Allocate a new disk block <DCA9>
D671 error? >>D6AA
D673 Store it in key pointer of entry (F030)
D679 and in BLKNUM for I/O
D67D Write zeroed (or DIR HDR) key block <DDDDD>
D680 error? >>D6AA
D682 Bump parent's file count (F013)
D68A Go update directory <D6AB>
D68D error? >>D6AA
D68F Checkpoint Volume Bit Map and exit >>D66E

D692 ********* POINT $48/49 AT DIRECTORY ENTRY ***********

D692 $48/$49 --> Entry
D696 Skip link pointers (+4)
D698 File entry number counter (F01E)
D69B ---
D69C Skip to proper entry
D69F Add entry length (F011)
D6A4 (bump MSB)
D6A8 (store LSB)
D6AA RETURN

ProDOS MLI -- V1.0.1 -- 1 JAN 84

ADDR DESCRIPTION/CONTENTS

D6AB *********** UPDATE DIRECTORY(S) **************

D6AB System date available? (BF90)
D6AB no, forget it >>D60B
D6B2 yes, copy to last modified date field (BF90)
D6B8 turn on HUBIT (backup) if appropriate (F03D)
D6C4 set DEVNUM of parent (F019)
D6CA and BLKNUM (F01C)
D6D4 reread DIR block containing entry <DDE1>
D6D7 error? >>D6AA
D6D9 Point to proper entry in buffer <D692>
D6D0 Copy constructed entry to buffer (F01F)
D6DB Is this block the DIR HDR block?
D6D0 no, write back new entry <DDDD>
D6DF error? >>D6AA
D705 and then read DIR HDR block <DDE1>
D708 error? >>D6AA
D70A in any case...
D70C copy back update file count to HDR (F013)
D715 and ACCESS byte (with Backup) (F010)
D71B write back HDR block <DDDD>
D71E HDR is done with it now...
D729 is this the VOL DIR? (F604)
D727 yes, all done -- exit >>D796
D729 no, subdirectory.. (F627)
D72C get parent pointer
D733 get parent entry no.. (F629)
D739 and entry len (F62A)
D73F read parent DIR block <DDE1>
D742 find entry for this subdirectory <D692>
D747 system date available? (BF90)
D74A no >>D759
D74C yes...
D750 copy system date/time to... (BF90)
D753 modified date/time in entry
D759 write it back <DDDD>
D75C error? >>D778
D768 BLKNUM = HDR block number
D769 same block we have now?
D76D yes, go back and date stamp >>D720
D76F no,
D773 read HDR block <DDE1>
D776 and go back to date stamp parent DIR >>D720
D778 error? then exit
ProDOS MLI -- V1.0.1 -- 1 JAN 84

D779 *********** NOT ProDOS VOLUME ERROR **********************

D779 ---
D77C RETURN

D77D *********** IS THIS ProDOS VOLUME? **********************

D77D Does previous block ptr = 0? (F600)
D78B no, not a ProDOS volume >>D779
D78D else, (F604)
D792 does VOL DIR'S STORAGE TYPE = $E or $F?
D794 no, error >>D779
D796 else, ok
D797 RETURN

D798 *********** GET FILE ENTRY **********************

D798 follow path to it's end <D7AB>
D79B error? >>D7AA
D7A0 copy file entry
D7A8 and exit
D7AA RETURN

D7AB *********** FOLLOW PATH TO A FILE **********************

D7AB get base dir's data <D92F>
D7AE error? >>D802
D7B0 another subdirectory in the path? >>D7DA
D7B2 no, at end of path (D82A)
D7B5 $48/$49 -- $P604 (HDR)
D7BD copy part of HDR to file entry
D7C7 file type = $F (Directory) (EPA8)
D7CA BLOCK = 2 (F01F)
D7CD No. blocks used = 4
D7CE ESF = $800
D7D2 TYPE = subdirectory ($D0)
D7D7 return to caller
D7D9 RETURN

*** SCAN DIRECTORY FOR FILE ***

D7DA indicate no free entry found as yet
D7DF signal in HDR block
D7E0 zero count of names examined
D7E5 find name in block <D880>
D7EB got it! >>D84F
D7EA not yet, how many entries expected? (F058)
D7ED less entry no. I just searched (F057)
D7F2 more file entries left to search? >>D804
D800 no, directory error

ProDOS MLI -- V1.0.1 -- 1 JAN 84

D802 ---
D803 RETURN

D804 yes, update entries left counter (F058)
D80A back to first buffer page (F49)
D80C check next block pointer (F602)
D814 if zero, directory error >>D800
D816 BLKNM: next directory block
D81D read next block <DDE1>
D820 no error, loop back for more >>D7E0
D822 exit if error

*** no MORE FILE ENTRIES ***

D823 free entry found in directory? (F05B)
D826 yes >>D843
D828 no, check pointers (F602)
D82B is there another block after this one? >>D832
D830 no... >>D843
D832 yes, free entry will be... (F01C)
D838 first in that block
D840 indicate free entry available (F05B)
D844 find next index name <D970>
D846 exiting with error
D847 no more indicies in path, file not found >>D84C
D849 else, path not found
D84B RETURN

D84C file not found error
D84E RETURN

*** FOUND FILE ENTRY ***

D84F advance to next subdir in path <D969>
D852 end -- save entry no. and exit >>D8C0
D856 get type of entry
D85A sub??
D85C no, exit path then >>D846
D860 copy key block no...
D862 to BLKNM
D865 and to current DIR block no (F81A)
D86F go read key block of subdirectory <DDE1>
D872 error? >>D898
D877 new file count (F058)
D880 check minimum version (F621)
D883 too new? >>D896
D88B count bits in reserved field of DIR hdr
D88C --- >>D888
D88F ---
D892 there must be 5 bits on (normally $75)
D894 (these are) >>D89A
file format

---

ProDOS MLI -- V1.0.1 -- 1 JAN 84

ADDR   DESCRIPTION/CONTENTS

D91C  skip to next entry (F05A)
D92B  end of block? if so, exit >>D91B
D926  bump $48/$49 by entry len
D92D  and go check next >>D8E5

D92F  GET DIRECTORY DATA

D92F  find base directory <D988>
D932  error? >>D987
D938  zero out my variables (F006)
D93E  set up device number (BF30)
D944  copy DIR HDR to my variables <D8A8>
D94D  copy TOTAL BLOCKS from VCB (F212)
D953  copy BIT MAP Pointer from VCB (F21A)
D959  copy Block No. of this directory (F046)
D95F  make second copy of file count (F013)
D969  advance to next subdir in path <D978>
D96C  and update index (F07A)
D96F  RETURN

D970  ADVANCE TO NEXT DIR NAME

D970  get this DIR's index (F07A)
D977  add len of name to move index to next name (F07A)
D97B  still in prefix portion? >>D983
D97D  no, now starting callier's path suffix (BF30)
D980  save last DEVNUM accessed (F05E)
D983  return with len of next dir in path (F100)
D987  RETURN

D988  FIND BASE DIRECTORY

D988  ---
D98A  get old PFIXPTR (BF9A)
D98D  fully qualified pathname? (F07C)
D990  no >>D993
D992  yes, no old PFIXPTR anymore
D993  save old prefix index (F07B)
D996  DEVNUM=0 (BF30)
D999  ---

*** SCAN VCB'S FOR A MOUNTED VOLUME ***

D99B  scan (F200)
D99E  got one >>D9AC
D9AE  else, bump to next VCB
D9A9  no mounted vols? remount them >>D9FD
** FIND LAST DIR IN PREFIX OR VCB **

D9AC store name length (F078)
D9AF same name as in path name? (F100)
D9B2 no -- skip it >>D9A0
D9C0 save VCB index (F051)
D9C3 DEVNUM = VCB's unit no. (F210)
D9C9 BLOCK = 2 (read VOLDIR if no old prefix)
D9D1 get old prefix index (F07B)
D9D4 ---
D9D5 accumulate a new index (F07A)
D9D8 no previous prefix >>D9E4
D9DB find last name in prefix (F100)
D9E0 read prefix directory instead of vol
D9EA read block <DDE1>
D9ED error? >>D9F5
D9F5 is this the right directory? <DA91>
D9F2 no >>D9F5
D9F4 yes -- exit!

*** IF NOT THERE, REMOUNT ALL VOLS ***

D9F5 open files? (F051)
D9FB yes, give up now >>DA16
D9FD else, (F07B)
DA0B put back old prefix length (F07A)
DA03 copy DVCLST from global page <DA57>
DA09 use last device accessed first >>DA0B
DA0B if none, get last in my device table
DA16 volume not found error
DA19 RETURN

DA1A ---
DA1D search for device in device table (DA)
DA25 when found, make it active device (BF31)
DA2A remove it from table (F08A)
DA2D find its VCB <DA69>
DA30 not found? >>DA56
DA32 volume mounted there? (F051)
DA38 no >>DA3F
DA3A yes, open files here? (F211)
DA3D yes, skip it -- get next unit >>DA0B
DA3F else, 
DA41 BLKNUM = 2 (vol dir)
DA47 read volume directory <DDE1>
DA4A error? >>DA0B
DA4C mount volume on VCB <DA87>
DA4F error? >>DA0B
DA51 is this his chosen volume? <DA91>

** NEXT OBJECT ADDR: D9A9 **
ProDOS MLI -- V1.0.1 -- 1 JAN 84

ADDR DESCRIPTION/CONTENTS

DAB7 ********** MOUNT NEW VOLUME **************************************

DAB7' volume mounted? (F051)
DAB8 no, continue >>DAC4
DAB9 if same one as one wanted? <DB1C>
DAC2 if so exit, else fall thru >>DB1B

DAC4 ********** SET UP VCB FROM VOLDIR ********************************

DAC4 zero out VCB
DAC5 is this a ProDOS volume? <D77D>
DAC2 no -- exit >>DB1B
DAC4 duplicate vol in VCB's? <DB3D>
DAC7 yes -- exit with that one instead >>DB1A
DAC9 get new volume's name length (F604)
DAE9 add to VCB index (F051)
DAE4 and copy to VCB name field in empty VCB (F604)
DAEF store in VCB name len field (F200)
DAF2 copy DEVNUM to VCB unit field (BF30)
DAF8 copy total blocks to VCB (F629)
DB04 copy block no. of vol dir to VCB
DB06 copy bit map block no. to VCB (F627)
DB1A exit
DB1B RETURN

DB1C ********** COMPARE VOL NAMES TO MAKE ****************************

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

DB1C get length (F604)
DB21 same in VCB? (F200)
DB24 no >>DB34
DB27 yes, add len to VCB index to point at (F050)
DB2A last char of name in VCB (F050)
DB31 compare names (F200)
DB34 SEC if no match
DB38 CLC if match
DB3C RETURN

DB3D ********** LOOK FOR DUPLICATE VOL *******************************

DB3D start with first VCB
DB3F --
DB40 this VCB has same name? <DB1C>
DB43 no >>DB54
DB45 yes, files open? (F211)
DB48 yes >>DB3E
DB4C no, mark VCB empty (NAME=0) (F200)
DB4F (UNIS=8) (F210)
DB52 and exit with no error >>DB5C
DB54 else,

ProDOS MLI -- V1.0.1 -- 1 JAN 84

ADDR DESCRIPTION/CONTENTS

DB56 bump to next VCB
DB5A and loop >>DB3F
DB5C exit no errors
DB5D RETURN

DB5E save flag (F075)
DB61 and VCB index of duplicate vol (F076)
DB64 exit with error
DB65 RETURN

DB66 ********** SEE IF A QUANTITY OF FREE ****************************

***** BLOCKS IS AVAILABLE ON VOL *****

DB66 any free blocks counted in VCB? (F051)
DB6F yes >>DBC3

*** COMPUTE VCB FREE BLOCK COUNT ***

DB71 no, how many bit map blocks are there? <DC15>
DB74 save it (less 1) (F05C)
DB79 zero scratch (will count free blocks) (F046)
DB7F no block found yet
DB84 checkpoint bit map buffer <DD86>
DB87 error? >>DBD7
DB8C BLKNUM = bit map pointer (F21A)
DB96 read block to buffer <<DE11>
DB99 error? >>DBD7
DB9B count free blocks marked <DBD8>
DB9E drop no. remaining to do (F05C)
DBA1 none left? >>DBAC
DBA3 some, BLKNUM = BLKNUM + 1
DBA9 go process that >>DB96

DBAC did we find a free bit? (F051)
DBB2 no -- volume full >>DBD4
DBB4 save VCB bitmap block offset (F21C)
DBB7 save free block count in VCB also (F047)
DBC3 are there enough to satisfy request? (F214)
DBD2 yes, exit
DBD3 RETURN

DBD4 volume full error
DBD7 RETURN

DBD8 ********** SCAN AND COUNT BITMAP BLOCKS ********************

DBD8 scan through both buffer pages
DBDF counting one bits <DC05>
DBEA
DBED found free block already? (F05B)
DBF0 if so -- done >>DC04
Beneath Apple ProDOS Supplement

ProDOS MLI -- V1.0.1 -- 1 JAN 84 NEXT OBJECT ADDR: DBF2

ADDR DESCRIPTION/CONTENTS

DBF2 any blocks found yet? (F046)
DBF8 no >>DC04
DBFA yes, compute total no. of bitmap blocks <DC15>
DBFC less number remaining (F05C)
DC01 gives bitmap block with first free bit (F05B)
DC04 exit

DC05 ******** COUNT ONE BITS IN A BYTE ***************

DC05 shift and...
DC08 count bits that are on (F046)
DC10 exit when byte goes to zero
DC14 RETURN

DC15 ******** COMPUTE NO. BITMAP BLKS -1 ***************

DC15 get blocks on vol count (-1) (F051)
DC21 --
DC22 isolate top nibble of block count
DC23 for bit map block count
DC26 RETURN

DC27 ******** FREE A BLOCK ON DISK ***************

DC27 save MSB (F05C)
DC2A and LSB
DC28 block number passed too big for (F213)
DC30 volume size? (F05C)
DC35 yes, error >>DC15
DC38 no, get bit position for block no.
DC3E save it (F05B)
DC42 divide block no. by 8 (F05C)
DC45 giving byte offset as remainder
DC48 save byte offset (F062)
DC51 make quotient/2 into block index (F05C)
DC54 remember which page in that block (F064)
DC57 read block map (after checkpoint) <DD57>
DC5A error? >>DC4A
DC5C are we at proper block of bitmap yet? (F069)
DC62 yes! >>DC7A
DC64 no -- checkpoint <DD86>
DC67 error? >>DC4A
DC69 indicate block wanted in VCB (F05C)
DC72 DEVNUM of bitmap (F066)
DC75 read actual block directly <DD97>
DC78 error? >>DC4A
DC7A get byte offset into page (F062)
DC7D which page? (F064)
DC80 get bit pattern to set (F05B)
DC83 page 0 >>DCBD
DC85 no, turn bit on in page 1 (F060)

******** GET NEXT BITMAP BLOCK ************
ProDOS MLI -- V1.0.1 -- 1 JAN 84
------------- NEXT OBJECT ADDR: DD4D

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>DD4D</td>
<td>checkpoint old one &lt;DD86&gt;</td>
</tr>
<tr>
<td>DD50</td>
<td>go read block &gt;&gt;DD57</td>
</tr>
<tr>
<td>DD53</td>
<td>disk full error</td>
</tr>
<tr>
<td>DD56</td>
<td>RETURN</td>
</tr>
</tbody>
</table>

DD57 ******* READ BITMAP BLOCK **********

DD57 have we read bitmap for this unit yet? (F051)
DD60 yes >>DD70
DD62 no, checkpoint bitmap of other unit <DD86>
DD65 error? >>DD85
DD6A get new bitmap unit no. (F210)
DD70 was bitmap modified? (F065)
DD73 yes >>DD7A
DD75 no, read it <DD97>
DD78 error? >>DD85
DD7A save bitmap block offset times 2 (F051)
DD7D (page number) (F21C)
DD84 exit
DD85 RETURN

DD86 ******* CHECKPOINT VOLUME BITMAP **********

DD86 --
DD87 needs checkpoint? (F065)
DD8A no >>DD85
DD8C yes, write it <DD99>
DD8F error? >>DD85
DD91 doesn't need checkpoint now
DD96 exit

DD97 ******* READ BITMAP **********

DD97 save DEVNUM (F066)
DD9A copy block offset wanted (F051)
DDA4 BITMAP BLOCK = BITMAP PTR + BLOCK OFFSET (F21A)
DDB2 set up read command

*** READ OR WRITE BITMAP ***

DD64 save I/O command
DDDA device = bitmap device (F066)
DDC0 block = bitmap block (F067)
DDCA point to bitmap buffer (D08F)
DDCD do the I/O <DD88>
DDDE restore old DEVNUM (BF30)
DDD5 ok? >>DDD8
DDD7 no, error exit
DDD8 RETURN

ProDOS MLI -- V1.0.1 -- 1 JAN 84
------------- NEXT OBJECT ADDR: DDD8

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDD9</td>
<td>******** WRITE BITMAP</td>
</tr>
<tr>
<td>DDDD</td>
<td>set up write command</td>
</tr>
<tr>
<td>DDDF</td>
<td>and go do it &gt;&gt;DD84</td>
</tr>
</tbody>
</table>

DDE8 ******* WRITE BLOCK ****

DDEC set up write command
DDE8 set unit to do I/O on I/O flag
DDE8 do block I/O <DDDA> (BF30)
DDE8 error? >>DDE5
DDE2 no errors, restore things and exit
DDE4 RETURN
DDE5 error exit
DDE7 RETURN

DDE8 ******* MLI GET MARK C **********

DDE8 copy mark to caller's |
DDE9 exit with no errors, list from FCB (F052)
DDE9 RETURN

DDE1 ******* MLI SET MARK C **********

DDE1A bad position error
DDE1D RETURN

DDE1E ******* MLI SET MARK C **********

DDE1E
DEB6 set up to...
DEB7 copy user's mark to temporary
DEB8 new mark variable (F066)
DEB9 make sure it will not exceed EOF (F315)
DEB10 else, error >>DEB1A
DEB11 *** STILL IN SAME DATA BLOCK? ***
DEB12 get old mark (F052)
DEB13 find its block no. (*2) (F31J)
DEB14 compute distance in pages from old mark's (F06B)
DEB15 block to new mark (F046)
DEB16 earlier -- need new data block >>DEB61
DEB17 too far forward -- need new block >>DEB61
DEB18 MSB's match? (F314)
DEB19 then mark is still in this block >>DEB7C
DEB20 check storage type (F037)
DEB21 zero? >>DEB60
DEB22 sapling, sapling or tree?
DEB23 no, special handling for DIR files >>DFAE
DEB24 stomp on PCB2's mark?? (F300+S52)
DEB25 (this should never happen anyway) (F300)
DEB26 and return with bad REFNUM error
DEB27 *** NEED DIFFERENT DATA BLOCK ***
DEB28 copy storage type (F037)
DEB29 old data block needs writing? (F308)
DEB30 no >>DEB8
DEB31 yes, do so <E087>
DEB32 error? >>DEB1
DEB33 see if new mark is outside the range of (F052)
DEB34 the current index block (F314)
DEB35 yes >>DEB1A
DEB36 yes >>DEB1A
DEB37 no, same index block (F056)
DEB38 check storage type
DEB39 sapling or tree are ok >>DEB20
DEB40 *** SEEDLING ***
DEB41 seedling, check position (F06B)
DEB42 if position is outside of block 0...
DEB43 promote to sapling >>DFAE
DEB44 else, (F30C)
DEB45 go get key block (seedling data block) >>DF72

*** NEED TO CHANGE DATA BLOCKS ***
DEBAD does old index block need dumping? (F308)
DEBAD no >>DEC6
DEBAD yes, do so <E09B>
DEBAD error? >>DEB1
DEBAD check storage type (F056)
DEBAD tree file?
DEBAD yes >>DEB3
DEBAD no, sapling (F06C)
DEBAD is position in first index block?
DEBAD no, need master index, subindex and data >>DF39
DEBAD yes, first index, reset flags >>DF20
DEBAD is this a seedling?
DEBAD if so, see if in first block >>DEB6

*** SAPLING ***
DEBAD no, sapling, read its only index block <E02E>
DEBAD error? >>DEB1
DEBAD set block no. of index block
DEBAD and continue below >>DF20
DEBAD error exit
DEB72 RETURN

*** TREE FILE/NEED ANOTHER INDEX BLOCK ***
DEBAD reset flags >>DF20
DEBAD read master index block <E02E>
DEBAD error? >>DEB1
DEBAD make index into block from (F06C)
DEBAD MSB of position/2
DEBAD is there a subindex there?
DEBAD yes! >>DF13
DEBAD no, fall thru to make one

*** GET NEW INDEX BLOCK ***
DEBAD need an index and data block
DEBAD go allocate them >>DF39
DEBAD set up block no. of subindex
DEBAD read it <E010>
DEBAD error? >>DEB1

*** SAPLING/TREE - THIS INDEX BLOCK ***
ProDOS MLI -- V1.0.1 -- 1 JAN 84
NEXT OBJECT ADDR: DFAD

ADDRESS DESCRIPTION/CONTENTS

DFAD 42

DFAE ******* SET DIR FILE POSITION ****************************

DFB5 DIR file?
DFB6 yes! >>DFB7
DFB8 no, bad storage type error
DFB4 got to SYSSERR <DFB9
DFB7 else, get page distance (F046)
DFB9 make it into blocks (divide by 2)
DFB1 new position beyond old? (F06B)
DFC4 yes >>DFD4
DFC5 else, use previous mark
DFC8 copy to BLKNUM <DFE2>
DFCB error? >>DFE1
DFCD count it (F05A)
DFD0 more to skip? >>DFC6
DFD2 no, got it >>DF7C
DFD4 use next block pointer in DIR block
DFD6 copy to BLKNUM <DFE2>
DFD9 error? >>DFE1
DFDB count it (F05A)
DFE8 more to skip >>DFD4
DFE0 got it now! >>DF7C

*** COPY LINK TO BLKNUM ***

DFE2 copy block number link
DFE4 to BLKNUM
DFE7 if non zero,
DFE0 then go read block. >>DFF3
DFEF else, EOF error
DFF1 --
DFF2 RETURN

DFD3 ******* READ FILE BLOCK ****************************

DFD3 set block number to read
DFD7 store read I/O command
DFD9 read to $48/$49 buffer
DFDF read the block <EO54>
EO00 error? >>EO0F
EO05 copy block no. just read to FCB
EO0F exit

EO10 ******* READ SUB-INDEX BLOCK ****************************

EO10 set read I/O command
EO14 read to $48/$49 buffer
EO16 read the block <EO54>
EO19 error? >>EO29
EO1E save BLKNUM in FCB as current index
**Beneath Apple ProDOS Supplement**

---

**ProDOS MLI -- V01.0.1 -- 1 JAN 84**

---

**ADDR** **DESCRIPTION/CONTENTS**

---

E020 block. (F30E)

E029 exit

---

**E02A ******** WRITE KEY INDEX BLOCK ***********************

**E02A set write I/O command**

**E02C and go do the I/O >>E030**

---

**E02E ******** READ KEY INDEX BLOCK ***********************

**E02E set read I/O command**

**E030 common code, save command**

**E033 block no. is key block in PCB (F052)**

**E038 use $48/$49 buffer**

*** I/O BLOCK ***

**E03A set I/O command**

**E03C and block no. (F30E)**

**E046 must be non-zero block number**

**E04A or horrible death!**

**E04F fall through to read/write block (F301)**

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

**E054 (xreg = buff ptr in zero page)**

**E055 disable**

**E056 set up buffer pointer**

**E061 get DEVNUM from PCB (F301)**

**E067 set I/O transfer has occurred flag**

**E06C set unit no. from DEVNUM (E3F0)**

**E071 no errors have occurred yet**

**E076 do block I/O <D06A>**

**E079 error? >>E07E**

**E07B no, exit normally**

**E07D RETURN**

**E07E else, exit with error**

**E080 RETURN**

---

**E081 ******** CHECKPOINT BITMAP & KEY BLOCK ***********************

**E081 checkpoint bitmap buffer <DD86>**

**E084 go write key block for file >>E02A**

---

**E087 ******** CHECKPOINT DATA BLOCK BUFFER ***********************

---

---

**ProDOS MLI -- V01.0.1 -- 1 JAN 84**

---

**ADDR** **DESCRIPTION/CONTENTS**

---

**E087 buffer pointer at $4A/$48**

**E089 point to block no. in PCB**

**E091 go write buffer to disk <E03A>**

**E094 error? >>E088**

**E098 go turn off $40 flag in PCB and exit >>E0AF**

---

**E09B ******** CHECKPOINT INDEX BLOCK BUFFER ***********************

**E09B checkpoint volume bitmap <DD86>**

**E09E use $48/$49 buffer**

**E0A0 block no. is current index block in PCB**

**E0A6 set to write**

**E0A8 go write it to disk <E03A>**

**E0AB error? >>E088**

**E0AD no longer needs checkpoint**

**E0AF set flags accordingly (F052)**

**E0B8 and exit**

---

**E0B9 *************** CHECKPOINT INDEX BLOCK BUFFER ***********************

---

**E0B9 search path for file <D798>**

**E0BC found it? >>E0C2**

**E0BE no, bad path error**

**E0C0 exit >>E0C9**

**E0C2 else, see if PCB already open on file <E1A9>**

**E0C5 for write, if not, continue. >>E0CB**

**E0C7 else, file already open error**

**E0C9 *** E0CA RETURN**

**E0CB get PCB index (F052)**

**E0D1 free PCB found? >>E0D7**

**E0D3 no, all PCB's in use error**

**E0D6 RETURN**

**E0D7 zero out unused PCB**
Hex to free FCB (F052)
at we found one
p this FCB >>E1ED

file ID's to see if this FCB (F300)
on the requested file. (F018)
? >>E1ED

FCB already open on file (F057)
abled? (F309)
allow multiple open access to file >>E1ED
error exit

index to start of FCB
next FCB
p >>E1B4
one, exit normally

********************************************
* MLI READ CALL *****
********************************************
0 data buffer <E403>
quest length <E3E8>
cess
marks <E415>
cess permitted?
20A
cess error
read past EOF? >>E231
052)
= EOF - current_mark (F315)
early at EOF? (F09A)
OG
error
zero length request? (F09A)
OG
mark and exit >>E2EF
y check data buffer <EE6C>
? >>E22E
storage type for file <E40E>
d kind of file?
24B
file >>E3B1
E24B else, set mark (to read proper buffers) <DE3B>
E24E error? >>E22E
E250 set up buffer indexing <E306>
E253 move all that can be moved out of data buff <E330>
E256 newline or len=0: exit now! >>E239
E258 newline enabled? continue block by block >>E24B
E25A at least 1 block's worth left to be read? (F06E)
E25E if not, never mind >>E24B
E260 if so, store block count wanted (F06F)
E263 get FCB flags <E7FC>
E266 data block modified?
E268 yes, continue block by block for now >>E24B

*** FAST DIRECT READ ROUTINE ***

E26A signal no read occurred yet (F070)
E26D read directly into caller's data buffer
E275 set mark/read data block to caller's buff <DE3B>
E278 error? >>E238
E27A bump buffer pointer to next location
E27E drop length remaining by 512 bytes (F06E)
E284 bump mark (F06B)
E28C and mark's MSB as necessary (F06C)
E28F check if we are out of index block (F06C)
E295 drop counter of multi-blocks (F06F)
E298 and keep on >>E247
E29A end of multi-block read, put ptrs back <E3A3>
E29D more to read? (F06D)
E2A3 no, exit through finish-up >>E26F
E2A5 yes, conventional block by block read then >>E24B
E2A7 crossed index block? go do set mark >>E275
E2A9 make index block offset from mark (F06C)
E2B2 BLKNUM = next block in index block
E2B6 zero entry?
E2C0 if so, no direct read can occur until next (F070)
E2C3 set-mark/read >>E2C8
E2C5 get MSB of BLKNUM
E2C8 (put index ptr back)
E2CC finish setting BLKNUM MSB
E2CE if no read occurred within setmark, (F072)
E2D1 go back to setmark call >>E275
E2D5 disable
E2D6 do I/O to caller's buffer directly
E2DA do block I/O directly <D06A>
E2D2 error? >>E2E2
E2E8 go back for more >>E27A

*** ERROR CLEANUP ***

E2E2 --
E2E3 --
E2E4 set buffer ptrs/VCB <E3A3>
E2E6 --
E2E8 finish up I/O <E2EF
E2ED exit with error
E2EE RETURN

E2EF ******* I/O FINISH UP ****************************

E2F2 return actual length read in caller's list (F07A)
E303 and exit by setting new mark >>E3B

E306 ******* SET UP BUFFER INDEXING ********************

E30A ---
E30C back up pointer to data buffer by an
E30E amount equal to the LSB of the mark (F06A)
E30F (which makes indexing easier)
E315 newline mode enabled? (F31F)
E319 no, CLC >>E325
E31B yes, SEC
E31C copy newline mask (F071)
E31F and newline character (F30A)
E325 first char index is LSB of mark in YREG (F06A)
E32B $4C/$4D --> page containing mark
E32C request count LSB in XREG (F06D)
E32F exit

E330 ******* COPY FROM I/O BLOCK BUFF ********************

****** TO DATA BUFF*******

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

E330 ---
E331 partial page to move? >>E338
E333 no, any full pages left? (F06E)
E336 no, read complete >>E3A
E33E yes, drop MSB of request length (F06E)
E33F ---
E33C copy one byte $4C --> $4E
E341 end of requested chunk? >>E35E
E343 no, newline enabled? >>E373
E345 ---
E347 no, loop for more >>E33C
E349 end of page, bump pointers
E350 bump new mark (F06B)
E355 finished first page of block buffer?
E359 if so, continue >>E33C
E35C no, need another block from disk >>E38
E35E another page in request length? (F06E)0
E361 no >>E37D
E364 more in this block-page? >>E36C
E366 no, on last page of block?
E36A no >>E36F
E36C yes, drop request len by one page (F06E)
E36F back up to next byte again (R)
E370 go copy next page >>E343
E373 check for newline
E37B not it, never mind! >>E345
E37D else, were we done with page?
E37F no >>E38A
E380 yes, bump pointer
E382 and mark (F06B)
E38A set overflow flag (read completed) (R2)
E38E update mark LSB (F06A)
E392 bump request count if necessary
E393 update count LSB (F06D)
E399 point beyond data in caller's buffer
E3A1 ---
E3A2 and exit

E3A3 clean up after direct I/O ******
E3A3 restore caller's data buffer pointer
E3AE go set buffers/find VCB and exit >>D3E8

E3B1 directory file read ******
E3B1 set mark/read <E3B3>
E3B4 error? >>E3B5
E3B6 set up buffer indexing <E306>
E3B9 move data from I/O buffer <E330>
E3BC need next block? >>E3B1
E3BE no, finish up I/O <<E2EF
E3C1 ok? exit >>E3E3
E3C3 not ok. EOF error?
E3C6 no, out now >>E3B4
E3C8 yes, point beyond EOF anyway? <DF7C>
E3CB zero out data block I/O buffer <DF5A>
E3D3 dummy up an empty DIR block with previous
E3D6 pointer and no forward pointer in I/O bus (F310)
E3D8 buffer.
E47C ****************************
**** MLI WRITE CALL ****
***************************

E47C copy request length <E38E>
E480 copy file mark <E415>
E483 extend EOF if needed <E45E>
E487 write access enabled?
E489 yes >>E48F
E48B no, access error
E48F check status of this device <E64E>
E492 error? >>E4CF
E494 request length = 0? (F09A)
E49A no >>E49F
E49C yes, exit through finish-up >>E2EF
E49F find caller's data buffer <E403>
E4A2 check storage type
E4A4 if DIR file, error >>E48B
E4A6 set mark/read blocks <E35B>
E4A9 error? >>E4CF
E4AB get FCB flags <E7FC>
E4AE any new blocks needed?
E4B0 no >>E514
E4B2 yes, allocating them
E4B4 ---
E4B5 count number of blocks needed
E4B8 store number needed (F854)
E4BE see if the blocks are available <DB66>
E4C1 no, disk full >>E4CF
E4C3 yes, get FCB flags <E7FC>
E4C6 master index block needed?
E4C8 no >>E4D7
E4CA yes, go add it <E58F>
E4CD and go on if no errors >>E4E3
E4CF error,
E4D0 set new mark/EOF <E442>
E4D4 and finish I/O, exit with error >>E288
E4D7 check FCB flags again <E7FC>
E4DA need sub-index block?
E4DC no >>E4E3
E4DE yes, go do it <E5DA>
E4EF error? >>E4CF
E4E3 buy a new block for data <E62E>
E4E6 error? >>E4CF
E4EB get FCB flags <E7FC>
E4ED indicate index buffer changed
E4ED no new blocks needed now
**ProDOS MLI -- V1.0.1 -- 1 JAN 84**

---

**ADDR** | **DESCRIPTION/CONTENTS**
--- | ---

**E58F**

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

(MAKE A TREE FILE)

- **E58F** add higher level <E5E7>
- **E592** error? >>E5E6
- **E594** get storage type <E40E>
- **E597** tree?
- **E599** yes >>E5A0
- **E59B** no, add another level <E5E7>
- **E59E** error? >>E5E6
- **E5A0** buy another block <E62E>
- **E5A3** error? >>E5E6
- **E5A5** make offset into current index block (F06C)
- **E5A8** from current mark
- **E5A9** point index to new block (F046)
- **E5B9** also save as current data block (F052)
- **E5C3** checkpoint bitmap & key block <E081>
- **E5C6** error? >>E5E6
- **E5CB** zero out new index block
- **E5D2** ---
- **E5D9** and exit

---

**E5DA**

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

- **E5DA** check storage type <E40E>
- **E5DF** seedling? >>E5E7
- **E5E1** no, read key index block <E02E>
- **E5E4** and go add data block >>E5A0
- **E5E6** exit if error occurs

*** ADD A HIGHER INDEX LEVEL TO FILE ***

- **E5E7** buy a block <E62E>
- **E5EA** error? >>E62D
- **E5EF** save old key block number (F30C)
- **E5F7** make new block the key block (F30C)
- **E604** and current index block in PCB (F06F)
- **E60D** store pointer to old key block
- **E610** in first position of new index
- **E617** checkpoint bitmap and new key block <E081>
- **E61A** error? >>E62D
- **E61C** get storage type <E40E>
- **E621** upgrade it to next higher type (F070)
- **E624** indicate DIR entry needs update (F080)
- **E62D** exit

---

**E62E**

********** BUY A DISK BLOCK ********************

- **E62E** allocate a disk block <DCA9>
- **E631** error? >>E64D
- **E633** get PCB flags <E7FC>
- **E636** indicate DIR entry needs update
- **E63F** add 1 to blocks in use for file
- **E64C** ---
- **E64D** exit

---

**E64E**

********** DO STATUS IF NO I/O YET ********************

- **E64E** get PCB flags <E7FC>
- **E651** any buffers in use? (I/O activity)
- **E653** if so, assume it's ok >>E64C
- **E655** no, (F301)
- **E658** select new device (BF30)

*** STATUS CALL ***

- **E65B** Save Unit Number
- **E65D** Save Block Number on stack
- **E663** Indicate Status call
- **E667** Indicate Block 0
- **E66B** Go do I/O <D0DA>
- **E66E** Restore Block Number to original value
- **E676** Exit

---

**E677**

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


---

- **E677** check REF NUM
- **E67B** specific close? >>E6B2

*** CLOSE ALL OPEN FILES ***

- **E67D** no errors yet (F07E)
- **E682** store PCB index (F052)
- **E686** get its level (F31B)
- **E689** if below system LEVEL, skip it (BF94)
- **E68C** yes, skip it >>E6A3
- **E68E** no, active PCB? (F300)
- **E691** no >>E6A3
- **E693** yes, flush it and update directory <E714>
- **E696** error? >>E6E5
- **E698** no, close specific PCB <E6B7>
- **E69D** is this a close-all?
- **E69F** yes, ignore errors >>E6A3
- **E6A1** no, stop on error >>E6E5
- **E6A3** bump PCB index to next one (F052)
E71C zero out...
E721 validit
E724 error?
E726 is write close all error
E728 no, exit chbck REP NUM <D3C5>
E72D has a w>711
E730 yes >>Ea access allowed? (F309)
E732 no, <E78> >>E709
E735 does awrite occurred since last flush? (F31C)
E737 no, then>>739
E739 else, >>c
E73C has dbctgng need flushing anyway?
E73E no >>E77F write now >>E709
E740 yes, >>ct 2CB flags <<E7FC
E743 error? b buffer changed?
E745 get flag
E748 has indexpoint it <<E087
E74A no >>77y>>711
E74C yes, <<us again <<E7FC
E74F error? w buffer changed?
E751 ---
E758 copy fiindexpoint it <<E08B
E762 set DEV>>711
E765 BLKNUM:
E76F read D't identifier data to my variables (F300)
E772 error? NUM (BF30)
E774 copy dir->current DIR block (F01A)
E777 are we in block <<DE1>
E780 no >>E77E>>711
E785 yes >>Fctory header <<DE0A>
E787 no, setc: block with this file's entry? (F01C)
E78B read ity?
E78E point c>>708
E791 copy fl new block number
E797 copy bl <<DE1>
E7A5 copy nel directory entry in block <D692>
E7B0 and next entry from directory <D79D>
E7BA isolatccks used count to entry (F318)
E7C3 combinw EOF (F315)
E7CB and up clo key block no. (F38C)
E7E1 write & new storage type (F385)
E7D1 error? it with name length (F01F)
E7D6 turn oface type/len field in entry (F01F)
E7DC same bctry back to directory <<DEAB
E7E4 no, exs >>7ED
E7EB yes, c'"write occured" flag (F31C)
E7EC RETURN; it now >>E7EB
\check point it also <<DD86
\rs, exit

COS Suplement
CLOSE ERROR

E7ED is this a close or flush all?
E7F2 no >>E7FA
E7F6 yes, save error code (F07E)
E7F9 RETURN

E7FA else, real error right now
E7FB RETURN

GET FCB FLAGS

E7FC load FCB flags (F052)
E7FF from FCB (F309)
E802 and exit

FILE ACCESS ERROR

E803 exit with file access error code
E806 RETURN

MLE SET_EOF CALL

E807 get storage type <E46E>
E80A if DIR file...
E80C its an access error >>E803
E80E else, save type for truncate to
E80F mess with.
E815 write access permitted? (F309)
E81A no, error >>E803
E81C check device status <E64E>
E81F error? >>E803
E829 copy EOF from FCB (F315)
E836 copy caller's new EOF
E841 compare old EOF to new (F04A)
E847 if less than or equal to... >>E84E
E849 if greater... >>E863

OLD EOF <= NEW EOF ***
NO TRUNCATE NEEDED ***

E84E new EOF beyond old
E855 copy caller's EOF to FCB
E860 exit by indicating flush needed >>EC59

flush first <E71C>
E896 error? >>E806
E8A8 $43/$49 -- end of data I/O buffer
E872 compare current mark to new EOF (F052)
E87F it is prior to EOF >>E898
E887 if past EOF, force mark back to EOF (F052)
E89B construct EOF block number and (F06A)
E89F byte offset into block from new EOF
E89E EOF mark, (F06B)
E896 on a block boundary? (F087)
E899 yes >>E888
E89B no, (F085)
E89F decrement block by 1
E8CD but don't let it fall below 0
E80B copy key block number (F052)
E867 set blocks freed to zero
E86F truncate file at new EOF <EC62>
E8F2 save status
E8FA set new key block in FCB (F07F)
E800 drop FCB block count by number (F318)
E803 of blocks freed in truncate routine. (F082)
E810 copy new storage type (F081)
E82D turn off all block allocation flags <DFA2>
E820 update VCB free block count <EB0D>
E82A copy mark (F312)
E832 force current mark to infinity (F312)
E839 go set mark <DE3B>
E83C no errors? >>E945
E83E if error, indicate in saved status
E844 but continue
E845 copy caller's EOF to FCB <E84E>
E84A exit ?

MLE GET_EOF CALL

MLE NEW_LINE CALL

MLE CLEAR_EOF CALL

MLE INIT_EOF CALL
Beneath Apple ProDOS Supplement

ProDOS MLI -- V1.0.1 -- 1 JAN 84

---

E95D ---
E95F copy newline mask
E968 and newline character
E96E return, no errors

E96F

*********** MLI GET FILE INFO ***********
*********** MLI GET FILE INFO ***********

E96F get the file entry <D798>
E972 ok? >>E9B6
E974 no, bad path?
E977 no, real error >>E9D3
E979 else, make it VOL DIR type
E97B with name length = 0 (F01F)
E980 no free blocks needed (F054)
E986 go through the motions to update the (F051)
E989 VCB block count. <DB71>
E99F copy blocks free from VCB (F215)
E99B copy total motions on volume to AUX ID (F213)
E9A4 total - free = block used (F054)
E9B6 shift type down from high nibble (F01F)
E9C2 copy the data to caller's paralist (EFCC)
E9D3 and exit

E9D4

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

E9D4 get the file entry <D798>
E9D7 error? >>E9FF
E9D9 indicate backup needed now (F055)
E9DB copy 13 params from caller's list to (EFCC)
E9DD file entry staging area >>E9F2
E9F2 ---
E9F7 if any spurious access bits are on...
E9FB access error!
E9FE RETURN

E9FF else, anything in his modification date?
EA03 no >>EA08
EA15 yes, go update directory >>D6BB
EA08 no, use system date then update directory >>D6AB

ProDOS MLI -- V1.0.1 --

---

ADDR DESCRIPTION/CONTENTS

EA0B "*********** MLI RENP *******"

EA0B follow path to f:
EA0E ok? >>EA14
EA10 no, bad name?
EA12 no, real error >:

*** RENAME VOLS ***

EA14 yes, copy new name
EA17 error? >>EA2C
EA19 get first length
EA1D get next (F100)
EA20 bad path if more
EA25 files open on volume
EA28 no, continue >>EA2B
EA2A yes, file open error
EA2C ---
EA2D RETURN

EA2E make type/len for file <D7AB>
EA35 write new name to <D7AB>
EA38 error? >>EA33
EA3F copy new name to EA2C
EA4B exit, no errors
EA4C RETURN

*** RENAME FLS a <EB35>

EA4D get path index <F100>
EA50 copy old name with
EA5C copy new name to
EA5F error? >>EA4D
EA61 get path index <EB2E>
EA67 compare all levels
EA6F save indies in
EA72 final name. (F07)
EA75 ---
EA77 a VOL DIR HDR
EA7F exit if they match
EA80 RETURN device's VCB (F100)

EA81 index to differ
EA84 point past it (F100)
EA8C must be the last
EA8F it isn't >>EA17
EA91 it is, (F07A)
EA94 do the same with the prefix to my buffer (F100)

<EB35>

F100 buffer of names up to and (F060)
F100 first which
F100 sees which point to (F079)
OS Supplement

1.0.1 -- 1 JAN 84

ION/CONTENTS

A file is only in last index? >>EAA5
path error

bad, follow path to new file <D7AB>
et an error >>EAA5
a duplicate name in directory

better be file not found
its really an error... >>EAA3
pathname again <D27F>
file entry <D798>
>EAA3
CB's <E1A9>
The file is open for write >>EAA3
ESS permit rename? <<

len from entry (F01F)

>>EADD
is sapling or tree?
>>EADD
compatibility error
path again <EB35>
>EAA3
path of last name (F079)
and name to file entry buffer (F100)
new len with type (F100)

update entry and exit >>EB23

block of this subdirectory <DDE1>
>EAA3
to directory's key block <EB26>
>EAA3
directory entry and exit >>DB88

COPY PATH TO BUFF & WRITE ******************************

/xlen and path to my buffer
the block >>DDDD

ProdOS MLI -- V1.0.1 -- 1 JAN 84

ADDR   DESCRIPTION/CONTENTS

EB35  ******  POINT TO NEW NAME  ***************
      COPY TO BUFFER

EB35  $48/$49 --> second pathname
EB40  go copy it >>D28A

EB43  ******  LOAD PATH INDEX  *********************

EB43  load pathname index
EB4A  (including prefix if any) (BF9A)
EB4D  --
EB4F  RETURN

EB50  ******  MLI DESTROY CALL  ******************
      ******************************

EB50  get file entry <D798>
EB53  error? >>EB9F
EB55  find FCB if any <E1A9>
EB58  FCB open? (F057)
EB5B  no >>EB61
EB5D  yes, file open error
EB60  RETURN

EB61  no free blocks needed
EB69  go compute VCB free block count <DB66>
EB6C  ok? >>EB73
EB6E  error, disk full?
EB71  no, real error >>EB9F
EB73  DESTROY enabled in ACCESS? (F03D)
EB78  yes >>EB7F
EB7A  no, access error
EB7F  check status of device (BF30)
EB85  error? >>EB9F
EB87  point to key block (F030)
EB96  get file?
EB9A  no >>EB60
EB9C  yes, handle differently >>EBF8

EB9F  RETURN

*** DESTROY NON-DIRECTORY FILE ***

EBA0  set new storage type (F081)
EBAA  zero EOF mark (F081)
EBAD  byte offset = $200
EBB2  free all blocks in file <EC62>
EBB5  error? >>EB9F
EBBB  free key block of seedling (F080)
ProDOS MLI -- V. 1.0.1

6.1 -- 1 JAN 84

NEXT OBJECT ADDR: EBC0

ProDOS MLI -- V. 1.0.1

CON/CONTENTS

ADDR   DESC

--

EBC0   error? 8 entry free
EBC2   mark DIR file block (F013)
EBC7   decrement vol bitmap <DD06>
EBD2   checkpos >> EB2? 8
EBD5   error? free block count in VCB <EBD0>
EBD7   update update the directory >> DAB0
EBDA   and go

EC62   check:
EC65   seed
EC67   yes
EC69   no
EC6B   yes
EC6D   no
EC6F   yes
EC71   no
EC74   go to:
EC77   go to:

EC7A   truncate
EC7C   at m
EC7F   read
EC82   error
EC84   at m
EC8A   yes

---

EBF0   add block (F082)
EBF0   in VCB, set search for free blocks at
EBF4   start n = bitmap (F21C)
EBF7   exit

FREE DIRECTORY FILE ***

EBC8   DIR file >> EC4B
EBFA   no, error = 0
EBFC   read vo >> EC4A
EBFF   error? = key block pointer (F030)
EC01   BLKNUM <DD06>
EC0B   read << EC6A
EC0E   errors? is any files ... (F625)
EC10   if DIR error
EC11   access to block marking entry free (F604)
EC1F   write b >> EC4A
EC25   error? = pointer is zero ... (F602)
EC27   if "next" end pretend it's a seeding >> EBB7
EC31   go back > cr
EC33   else ... (next block << EC27)
EC36   free n >> EC4A
EC39   error? = next block (F602)
EC3B   BLKNUM >> <D31E>
EC45   read it; continue in loop >> EC27
EC48   if ok, error exit
EC4A   else, nible file format error
EC4B   incompa

EC50   SET WAIT OCCURED FLAG **************

E51   save << write occurred (F052)
EC53   indicate regs and exit
EC5E   restrt
EC61   RETURN
ECD8 now go
ECD4 which: free all the sub-index blocks (F084)
ECD7 error? follow EOF <ED2A>
ECD9 write? >ECDF
ECD6 error? back master index <DDDD>
ECD8 EOF in >>ECDF
ECF1 if so, first subindex? (F084)
ECF3 else, is demon to sapling file >>ED8
ECF6 count, BLKNUM = subindex block which (F080)
ECF8 (exit) has the EOF mark
ED02 else, if none there >>ECDE
ED05 and go, read subindex block <DDE1>
ED07 unless time below >>ED12
ED09 if there is an error
ED0B demon
ED0D error? tree to sapling <ED7E>
- - - - >ECDF
*** TRUNCATE SAPLING FILE ***
ED08 read file
ED10 error? block <ED71>
ED12 get list >>ECDF
ED16 if zero of block number (F085)
ED18 else, if no blocks are free >>ED22
ED1A follow free rest of blocks in index <ED2A>
ED1D write the EOF, check for error >>ECDF
ED20 error? index block back <DDDD>
ED22 get list >>ECDF
ED25 might? block number (F085)
ED27 no, get the block #? >>EDC3
ED2A from <BLKNUM of data block (F080)
ED2F (no subindex block)
ED36 read lock allocated? >>ECDE
ED39 and quota block <DDE1>
ED3B unless time below >>ED4B
- - - - error occurred
ED3C back to
ED3F no, get zero block #? (F084)
ED41 yes, >>D27
ED44 error? demon to seeding <ED7E>
- - - - >ED70
*** TRUNCATE SEEDLING FILE ***
ED46 read k
ED49 error? block <ED71>
ED4B first? >ED70
ED4E yes >>page? (F087)
ED51 no, back to
ED53 get byte be second >>ED6F
ED56 - - 10 offset (F086)

Pro DOS MLI

ADDR DSCR
ED58 zero
ED66 ini PROJ/CONTENT
ED6C the

ED6F ext beyond EOF mark (F700)
ED70 ext other pages if necessary (>EDD)
write block back and exit
ED71 *** normally
ED7B exit

ED7E *** READ KEY BLOCK *******
ED7F J = key block number (F07F)
ED87 error? >>ED1D
ED89 get *** DEMOTE FILE TO SMALLER FILE TYPE**********
ED96 redo
ED9E ext block (F080)
ED9F RT? >>ED9F
EDA0 ext block from old index (F081)
get storage type by one (F081)
exit
EDA0 ***
ED2A saves
ED28 for *** FREE ALL BLOCKS IN AN INDEX BLK **************
ED33 if
ED3A free
ED3D error BLKNUM
ED3F zero each index entry after mark, (F05D)
EDCB *** 1 non-zero...
EDC6 lock the block <DC27>
EDCC redo the index entry now (F0D)
ED6D and through all entries >>SA8
ED77 *** are old BLKNUM
ED77 *** exit
ED9D get
EDDC can *** ALLOCATE I/O BUFFER ***********************
EDDE else
ED00 can
ED28 else I/O buffer page number,
ED77 Q37 be below $800
ED80 must, error >>EE22
EE21 be above SBC00
EE22 check error >>EE22
EE27 PR? >>4B --> I/O buffer
EE2D be page aligned! >>EE22
EE03 if 0, each page of I/O buffer for <EE5D>
EE5D mark each page as allocated <EE5D>
Beneath Apple ProDOS Supplement

ProDOS MLI -- V1.0.1 -- 1 JAN 84

ADD DESCRIPTION/CONTENTS

---

EE66 in system memory bit map (BF58)
EE13 assign buffer number (REFNUM*2) in FCB (F300)
EE1B and save buffer location in buffer list
EE20 exit
EE21 RETURN
EE22 bad I/O buffer error
EE25 RETURN

EE26 ********** LOCATE I/O BUFFER **********

---

EE26 --
EE27 AREG contains buffer number *2 (BF6E)
EE2A move buffer pointer to NCTXBUF variable (F09D)
EE33 exit

EE34 ********** FREE I/O BUFFER **********

---

EE34 is buffer already free? <EE26>
EE39 yes, exit >>EE5B
EE3D zero its address in system global page (BF6F)
EE4A --
EE4B free each page in buffer <EE5D>
EE4E by marking system bit map
EE5B exit
EE5C RETURN

EE5D ********** LOCATE BIT MAP POSITION **********

---

(GIVEN PAGE NUMBER)

EE5D XREG contains page number
EE5E compute page number times 8
EE61 use as offset for bitmask (EFC0)
EE66 page number / 8 = byte offset
EE69 into bitmap
EE6B exit

EE6C ********** CHECK BUFFER VALIDITY **********

---

START > $200 END < $BF00

EE6C get buffer address (MSB)
EE70 must be >$200 else error >>EE22
EE72 get length ($F09B)
EE78 compute last page no. of buffer
EE7D
EE84 may not extend into $BF00
EE86 else, error >>EE22

---

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

EE89 ---
EE8A see if this page is allocated <EE5D>
EE90 if so, error >>EE22
EE92 else, check other page also
EE96 then exit if both have been checked
EE97 RETURN

EE98 ********** MLI GET BUFF CALL **********

---

EE98 get next available buffer
EE9D put its address in caller's parmlist
EEA5 and exit
EEA6 RETURN

EEA7 ********** MLI SET BUFF CALL **********

---

EEA7 mark his buffer allocated
EEAC error? >>EECE
EEAE get old buffer address ($09E)
EEB8 free old buffer's pages in map <EE43>
EEBF copy old buffer contents
ECC1 to new buffer
ECCD then exit
EECE RETURN

EECF ********** GO TO QUIT CODE HANDLER **********

---

EECF enable 2nd 4K bank of language card (C083)
EED2 (it lives at $D100-$D3FF) (C083)
EED5 save zero page $00 through $03 on stack
EE1 Set ($00) -> $100
EE3 Set ($02) -> $1000
EEEF Set Y = 0
EEF0 3 pages of code to copy
EEF2 ---
EEF3 copy quit code handler to $1000
EEF1 Restore zero page to original state
EF0D enable last 4K bank of language card (C080)
EF10 (MLI) (C088)
EF15 point RESET vector at $1000 (B3F2)
EF1D set power-up byte properly
EF22 go to quit code handler at $1000 >>1000
<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>EF25</td>
<td>DATA AREA</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>EF25</td>
<td>MLI COMMAND TABLE</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>IN HASH CODE ORDER: IF COMMAND IS...</td>
</tr>
<tr>
<td></td>
<td>ABCD EFGH (IN BINARY BITS)</td>
</tr>
<tr>
<td></td>
<td>INDEX IS COMPUTED AS:</td>
</tr>
<tr>
<td></td>
<td>0000 EFGH</td>
</tr>
<tr>
<td></td>
<td>+0000 ABCD</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>EF26 GET BUF</td>
</tr>
<tr>
<td></td>
<td>EF27 UNUSED</td>
</tr>
<tr>
<td></td>
<td>EF28 UNUSED</td>
</tr>
<tr>
<td></td>
<td>EF29 ALLOC INTERRUPT</td>
</tr>
<tr>
<td></td>
<td>EF30 READ BLOCK</td>
</tr>
<tr>
<td></td>
<td>EF31 WRITE BLOCK</td>
</tr>
<tr>
<td></td>
<td>EF32 GET TIME</td>
</tr>
<tr>
<td></td>
<td>EF33 EXIT</td>
</tr>
<tr>
<td></td>
<td>EF34 CREATE</td>
</tr>
<tr>
<td></td>
<td>EF35 DESTROY</td>
</tr>
<tr>
<td></td>
<td>EF36 RENAME</td>
</tr>
<tr>
<td></td>
<td>EF37 SET FILE INFO</td>
</tr>
<tr>
<td></td>
<td>EF38 GET FILE INFO</td>
</tr>
<tr>
<td></td>
<td>EF39 ON LINE</td>
</tr>
<tr>
<td></td>
<td>EF3A SET PREFIX</td>
</tr>
<tr>
<td></td>
<td>EF3B GET PREFIX</td>
</tr>
<tr>
<td></td>
<td>EF3C OPEN</td>
</tr>
<tr>
<td></td>
<td>EF3D NEWLINE</td>
</tr>
<tr>
<td></td>
<td>EF3E READ</td>
</tr>
<tr>
<td></td>
<td>EF3F WRITE</td>
</tr>
<tr>
<td></td>
<td>EF40 CLOSE</td>
</tr>
<tr>
<td></td>
<td>EF41 FLUSH</td>
</tr>
<tr>
<td></td>
<td>EF42 SET MARK</td>
</tr>
<tr>
<td></td>
<td>EF43 GET MARK</td>
</tr>
<tr>
<td></td>
<td>EF44 SET EOF</td>
</tr>
<tr>
<td></td>
<td>EF45 GET BUF</td>
</tr>
<tr>
<td></td>
<td>EF46 UNUSED</td>
</tr>
<tr>
<td></td>
<td>EF47 UNUSED</td>
</tr>
<tr>
<td></td>
<td>EF48 UNUSED</td>
</tr>
<tr>
<td></td>
<td>EF49 ALLOC INTERRUPT</td>
</tr>
<tr>
<td></td>
<td>EF4A DEALLOC INTERRUPT</td>
</tr>
<tr>
<td></td>
<td>EF4B UNUSED</td>
</tr>
<tr>
<td></td>
<td>EF4C UNUSED</td>
</tr>
<tr>
<td></td>
<td>EF4D READ BLOCK</td>
</tr>
<tr>
<td></td>
<td>EF4E WRITE BLOCK</td>
</tr>
<tr>
<td></td>
<td>EF4F GET TIME</td>
</tr>
<tr>
<td></td>
<td>EF50 EXIT</td>
</tr>
<tr>
<td></td>
<td>EF51 CREATE</td>
</tr>
<tr>
<td></td>
<td>EF52 DESTROY</td>
</tr>
<tr>
<td></td>
<td>EF53 RENAME</td>
</tr>
<tr>
<td></td>
<td>EF54 SET FILE INFO</td>
</tr>
<tr>
<td></td>
<td>EF55 GET FILE INFO</td>
</tr>
<tr>
<td></td>
<td>EF56 ON LINE</td>
</tr>
<tr>
<td></td>
<td>EF57 SET PREFIX</td>
</tr>
<tr>
<td></td>
<td>EF58 GET PREFIX</td>
</tr>
<tr>
<td></td>
<td>EF59 OPEN</td>
</tr>
<tr>
<td></td>
<td>EF5A NEWLINE</td>
</tr>
<tr>
<td></td>
<td>EF5B READ</td>
</tr>
<tr>
<td></td>
<td>EF5C WRITE</td>
</tr>
<tr>
<td></td>
<td>EF5D CLOSE</td>
</tr>
<tr>
<td></td>
<td>EF5E FLUSH</td>
</tr>
<tr>
<td></td>
<td>EF5F SET MARK</td>
</tr>
<tr>
<td></td>
<td>EF60 GET MARK</td>
</tr>
<tr>
<td></td>
<td>EF61 UNUSED</td>
</tr>
<tr>
<td></td>
<td>EF62 SET EOF</td>
</tr>
<tr>
<td></td>
<td>EF63 GET EOF</td>
</tr>
<tr>
<td></td>
<td>EF64 SET BUF</td>
</tr>
<tr>
<td></td>
<td>EF65 CREATE</td>
</tr>
<tr>
<td></td>
<td>EF66 DESTROY</td>
</tr>
<tr>
<td></td>
<td>EF67 RENAME</td>
</tr>
<tr>
<td></td>
<td>EF68 SET FILE INFO</td>
</tr>
<tr>
<td></td>
<td>EF69 GET FILE INFO</td>
</tr>
<tr>
<td></td>
<td>EF70 ON LINE</td>
</tr>
<tr>
<td></td>
<td>EF71 SET PREFIX</td>
</tr>
<tr>
<td></td>
<td>EF72 GET PREFIX</td>
</tr>
<tr>
<td></td>
<td>EF73 OPEN</td>
</tr>
<tr>
<td></td>
<td>EF74 NEWLINE</td>
</tr>
<tr>
<td></td>
<td>EF75 READ</td>
</tr>
<tr>
<td></td>
<td>EF76 WRITE</td>
</tr>
<tr>
<td></td>
<td>EF77 CLOSE</td>
</tr>
<tr>
<td></td>
<td>EF78 FLUSH</td>
</tr>
<tr>
<td></td>
<td>EF79 SET MARK</td>
</tr>
</tbody>
</table>

**EF45 PARAMETER COUNT TABLE**

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Pro DOS MLI -- V1.0.1 -- 1 JA

ADDR DESCRIPTION/CONTENTS

EF01 Blocks Used
EF03 End of File
EF06 Special ID (Must be INFO BYTE
EF07 'WUSTON!'

THE FOLLOWING IS

Pro DOS MLI -- V1.0.1 -- 1 JAN 84

ADDR DESCRIPTION/CONTENTS

EF00 File Type (Directory)
EF01 Block Number
EF02 Number of Blocks
EF03 End of File

EFC0 ********** BITMASK TABLE ***************

EFC0 10000000
EFC1 01000000
EFC2 00100000
EFC3 00010000
EFC4 00001000
EFC5 00000100
EFC6 00000010
EFC7 00000001

EFC8 ********** OFFSETS TO DATA AT IF300 ***********

EFC0 Key Block
EFC1 # Blocks Used
EFC2 End of File

EFC8 ********** SET/GET FILE INFO OFFSETS *************
F011  Entry Length
F012  Entries per Block
F013  File Count
F015  Bit Map Pointer
F017  Total Blocks
  THE FOLLOWING 6 BYTES UNIQUELY IDENTIFY
  A FILE:
F019  Device Number
F01A  Current Directory Block Number (HDR)
F01C  Block Number of File Entry in Directory
F01E  File Entry Number in Directory

F01F  *********** FILE ENTRY BUFFER ***********************

  F01F  Type/Length (TTTTTTTTTT)
  F020  File Name (Max 15) >>000F
  F02F  File Type
  F030  Key Pointer
  F032  Blocks Used
  F034  End of File
  F037  DateTime (Creation)
  F03B  Version
  F03C  Min Version
  F03D  Access Attribute
  F03E  Aux Type (Load Address/Record Length)
  F040  DateTime (Last Mod)
  F044  Header Pointer

F046  *********** Variable Work Area ***********************

  F046  3 Byte Scratch

  F049  ---

  F04A  End of File

  F04D  Previous Mark

  F050  Compare Vol Name Scratch
  F051  Offset into VCB Table ($F200)
  F052  Offset into VCB Table ($F300)
  F053  Free FCB found Flag

  F054  Number of Free Blocks needed

  F056  Storage Type
       Number of Entries Examined or...
  F057  FCB already open flag
  F058  File Count

  F05A  Entries/Block Loop Count/Free
  Free Entry Found Flag (if)

  F05B  bit for free
  F05C  # Blocks in Bitmap left to search
  F05D  $ Register temp
  F05E  Pathname Length
  F05F  Devnum for Prefix Directory
  F060  Block of Prefix Directory search
  F062  Bitmap Byte Offset in Page
  F063  Bitmap Page Offset
  F064  Bitmap Buffer Page (0 or 1)
  F065  Bitmap (if 1 or needs

  F066  Bitmap DEVDUM
  F067  Bitmap Block Number
  F069  Bitmap Block offset for multi

  New Mark to be positioned
  or New Moving Mark (for RE
  or New EOF for SET_EOF

  F06A  iblock Bitmaps

  F06D  Request Count (Read/Write #eco for Set Mark

  F06F  Multi-Block I/O count

  F070  Newline character
  F071  Newline mask
  F072  I/O Transfer occurred flag

  F073  MLI Command * 2

  F074  ORed into Access Flags ($28
  F075  Duplicate Volume Flag (if $32

  F076  Duplicate Volume's VCB index
  F077  MLI function code (low 5 bit

  F078  Characters in current path

  F079  ONLINE: volname len - loop (p)

  F07A  new pathname: index to last

  F07B  Old pathname: index to last

  F07C  ONLINE: index to data buffer ind x l v 1 or

  F07D  Old PFXPTR value

  F07E  Pathname fully qualified filename

  F08C  Pathname: temp save area

  F08D  ONLINE: DEVCNT

  F08E  close-all error code

  F08F  Set EOF: new Key Block pointer (if $FF

  F08A  New storage type (SET_EOF) for index or...

  F08B  Freed Blocks count

  F08D  EOF Block number (MSB then

  F08E  EOF byte offset into Block

  F08F  EOF - Master index counter

  F089  Save area for index into ..

  F08A  or old path name

  F08B  and last

  F08D  or NEW

  F08E  header

  F08F  and

  F089  area

  F08A  or old path name

  F08B  and last

  F08D  or NEW

  F08E  header

  F08F  and

  F089  area

  F08A  or old path name

  F08B  and last

  F08D  or NEW

  F08E  header

  F08F  and

  F089  area

  F08A  or old path name

  F08B  and last

  F08D  or NEW

  F08E  header

  F08F  and

  F089  area

  F08A  or old path name

  F08B  and last

  F08D  or NEW

  F08E  header

  F08F  and

  F089  area
Beneath Apple ProDOS

ProDOS MLI -- V1.0
---------
ADDR  DESCRIPTIC
---------

F099A ******DF
(also used
8 blocks)

F09A ******DF
---------

F092 length of
---------

device table
---------

F09D next buffer
---------

F09F 6 byte zero
---------

F0AF 16 byte index block EOF to keep track of
---------

F0AF jump vector to be freed at a time
---------

F0B7 not used
e page one
---------

F100X *******PM
---------

F104 prefix page savearea >>0006
---------

generic page savearea >>0006
---------

F100 pathname block
---------

F200X ******VOL
---------

VOLDATA AREA ******
---------

F200X Length (8 NAME) L2 NAME2 ... 0
---------

F201 File Name
---------

F210 Unit Number
---------

F211 Files Open may be used to use it,
---------

F212 Total Block Number to the pathname again.
---------

F214 Blocks Free
---------

F216 Block Number
---------

F218 not used
---------

F219 not used
---------

F2110 CONTROL BLOCKS ******
---------

F21A Bit Map Position
---------

Block of Net XD
---------

F21C next free block >>000F
---------

F21E Count of entries
---------

F220 VCB through
---------

at of Vol Dir Key Block
---------

F400X BITMAP BUFFER ******
---------

F400 Buffer 1st half >>0100
---------

F500 Buffer 2nd half >>0100
---------

F600X PRIMARY BUFFER ******
---------

(used for several things, VOL DIR HDR is
---------

mapped into it below)
<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>F600</td>
<td>Pointer Fields</td>
</tr>
<tr>
<td></td>
<td>&quot;*** VOLUME DIRECTORY HEADER ***&quot;</td>
</tr>
<tr>
<td>F604</td>
<td>Type/Length (TTTLLLLL)</td>
</tr>
<tr>
<td>F605</td>
<td>File Name (Max 15) &gt;&gt;0000</td>
</tr>
<tr>
<td>F614</td>
<td>Reserved &gt;&gt;0008</td>
</tr>
<tr>
<td>F61C</td>
<td>Creation Datetime</td>
</tr>
<tr>
<td>F620</td>
<td>Version</td>
</tr>
<tr>
<td>F621</td>
<td>Min Version</td>
</tr>
<tr>
<td>F622</td>
<td>Access Byte</td>
</tr>
<tr>
<td>F623</td>
<td>Entry Length</td>
</tr>
<tr>
<td>F624</td>
<td>Entries per Block</td>
</tr>
<tr>
<td>F625</td>
<td>File Count</td>
</tr>
<tr>
<td>F627</td>
<td>Bitmap Pointer</td>
</tr>
<tr>
<td>F629</td>
<td>Total Blocks</td>
</tr>
<tr>
<td>F62B</td>
<td>(remainder of first page of block) &gt;&gt;0005</td>
</tr>
<tr>
<td>F700</td>
<td>(second page of block) &gt;&gt;0100</td>
</tr>
</tbody>
</table>
### ProDOS System Global Page

#### Portions of this page of memory are rigidly defined by the MLI and are unlikely to move in later versions of ProDOS. However, some portions are less stable and could change in future releases.

<table>
<thead>
<tr>
<th>ADDR</th>
<th>LABEL</th>
<th>CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>BF00-BF02</td>
<td>ENTRY</td>
<td>Jump Vectors</td>
</tr>
<tr>
<td>BF00-BF05</td>
<td>JSPECIAL</td>
<td>JMP to system death code (via $BF6F).</td>
</tr>
<tr>
<td>BF06-BF08</td>
<td>DATETIME</td>
<td>JMP to Date/Time routine (RTS if no clock).</td>
</tr>
<tr>
<td>BF09-BF0B</td>
<td>SYSERR</td>
<td>JMP to system error handler.</td>
</tr>
<tr>
<td>BF0C-BF0E</td>
<td>SYSDEATH</td>
<td>JMP to system death handler.</td>
</tr>
<tr>
<td>BF0F</td>
<td>SERR</td>
<td>System error number.</td>
</tr>
<tr>
<td>BF10-BF11</td>
<td>DEVADR01</td>
<td>Slot 1, drive 1 device driver address.</td>
</tr>
<tr>
<td>BF12-BF13</td>
<td>DEVADR11</td>
<td>Slot 2, drive 1 device driver address.</td>
</tr>
<tr>
<td>BF14-BF15</td>
<td>DEVADR21</td>
<td>Slot 3, drive 1 device driver address.</td>
</tr>
<tr>
<td>BF16-BF17</td>
<td>DEVADR31</td>
<td>Slot 4, drive 1 device driver address.</td>
</tr>
<tr>
<td>BF18-BF19</td>
<td>DEVADR41</td>
<td>Slot 5, drive 1 device driver address.</td>
</tr>
<tr>
<td>BF1A-BF1B</td>
<td>DEVADR51</td>
<td>Slot 6, drive 1 device driver address.</td>
</tr>
<tr>
<td>BF1C-BF1D</td>
<td>DEVADR61</td>
<td>Slot 7, drive 1 device driver address.</td>
</tr>
<tr>
<td>BF1E-BF1F</td>
<td>DEVADR71</td>
<td>Slot 8, drive 1 device driver address.</td>
</tr>
<tr>
<td>BF20-BF21</td>
<td>DEVADR82</td>
<td>Slot 0 reserved.</td>
</tr>
<tr>
<td>BF22-BF23</td>
<td>DEVADR92</td>
<td>Slot 1, drive 2 device driver address.</td>
</tr>
<tr>
<td>BF24-BF25</td>
<td>DEVADRE2</td>
<td>Slot 2, drive 2 device driver address.</td>
</tr>
<tr>
<td>BF26-BF27</td>
<td>DEVADR32</td>
<td>Slot 3, drive 2 device driver address.</td>
</tr>
<tr>
<td>BF28-BF29</td>
<td>DEVADR42</td>
<td>Slot 4, drive 2 device driver address.</td>
</tr>
<tr>
<td>BF2A-BF2B</td>
<td>DEVADR52</td>
<td>Slot 5, drive 2 device driver address.</td>
</tr>
<tr>
<td>BF2C-BF2D</td>
<td>DEVADR62</td>
<td>Slot 6, drive 2 device driver address.</td>
</tr>
<tr>
<td>BF2E-BF2F</td>
<td>DEVADR72</td>
<td>Slot 7, drive 2 device driver address.</td>
</tr>
<tr>
<td>BF30</td>
<td>DEVNUM</td>
<td>Slot and drive (DSS$0000) of last device.</td>
</tr>
<tr>
<td>BF31</td>
<td>DEVCTN</td>
<td>Count (minus 1) of active devices.</td>
</tr>
<tr>
<td>BF32-BF3F</td>
<td>DEVLIST</td>
<td>List of active devices (slot, drive and identification--DSS$III).</td>
</tr>
<tr>
<td>BF40-BF4F</td>
<td>IrqIX</td>
<td>Copyright notice.</td>
</tr>
<tr>
<td>BF50-BF55</td>
<td>IRQIXT</td>
<td>Switch in language card and call IRQ handler at $FF08.</td>
</tr>
<tr>
<td>BF56-BF57</td>
<td>TEMP</td>
<td>Temporary storage for IRQ code.</td>
</tr>
<tr>
<td>BF58-BF6F</td>
<td>BITMAP</td>
<td>Bitmap of low 48K of memory.</td>
</tr>
<tr>
<td>BF70-BF71</td>
<td>BUFFER1</td>
<td>Open file 1 buffer address.</td>
</tr>
<tr>
<td>BF72-BF73</td>
<td>BUFFER2</td>
<td>Open file 2 buffer address.</td>
</tr>
<tr>
<td>BF74-BF75</td>
<td>BUFFER3</td>
<td>Open file 3 buffer address.</td>
</tr>
<tr>
<td>BF76-BF77</td>
<td>BUFFER4</td>
<td>Open file 4 buffer address.</td>
</tr>
<tr>
<td>BF78-BF79</td>
<td>BUFFER5</td>
<td>Open file 5 buffer address.</td>
</tr>
<tr>
<td>BF7A-BF7B</td>
<td>BUFFER6</td>
<td>Open file 6 buffer address.</td>
</tr>
<tr>
<td>BF7C-BF7D</td>
<td>BUFFER7</td>
<td>Open file 7 buffer address.</td>
</tr>
<tr>
<td>BF7E-BF7F</td>
<td>BUFFER8</td>
<td>Open file 8 buffer address.</td>
</tr>
</tbody>
</table>
### Interrupt Information

<table>
<thead>
<tr>
<th>ADDR</th>
<th>LABEL</th>
<th>CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>BF80-BF81</td>
<td>INTRPT1</td>
<td>Interrupt handler address (highest priority).</td>
</tr>
<tr>
<td>BF82-BF83</td>
<td>INTRPT2</td>
<td>Interrupt handler address.</td>
</tr>
<tr>
<td>BF84-BF85</td>
<td>INTRPT3</td>
<td>Interrupt handler address.</td>
</tr>
<tr>
<td>BF86-BF87</td>
<td>INTRPT4</td>
<td>Interrupt handler address (lowest priority).</td>
</tr>
<tr>
<td>BF88</td>
<td>INTAREG</td>
<td>A-register save area.</td>
</tr>
<tr>
<td>BF89</td>
<td>INTXREG</td>
<td>X-register save area.</td>
</tr>
<tr>
<td>BF8A</td>
<td>INTYREG</td>
<td>Y-register save area.</td>
</tr>
<tr>
<td>BF8B</td>
<td>INTSREG</td>
<td>S-register save area.</td>
</tr>
<tr>
<td>BF8C</td>
<td>INTPREG</td>
<td>P-register save area.</td>
</tr>
<tr>
<td>BF8D</td>
<td>INTBANKID</td>
<td>Bank ID byte (ROM, RAM1, or RAM2).</td>
</tr>
<tr>
<td>BF8E-BF8F</td>
<td>INTADDR</td>
<td>Interrupt return address.</td>
</tr>
</tbody>
</table>

### General System Info

<table>
<thead>
<tr>
<th>ADDR</th>
<th>LABEL</th>
<th>CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>BF90-BF91</td>
<td>DATE</td>
<td>YYYYMM MNMODD0DD.</td>
</tr>
<tr>
<td>BF92-BF93</td>
<td>TIME</td>
<td>HHHHMMMM/MMMM.</td>
</tr>
<tr>
<td>BF94</td>
<td>LEVEL</td>
<td>Current file level.</td>
</tr>
<tr>
<td>BF95</td>
<td>SBIT</td>
<td>Backup bit.</td>
</tr>
<tr>
<td>BF96-BF97</td>
<td>SPARE1</td>
<td>Currently unused.</td>
</tr>
<tr>
<td>BF98</td>
<td>MACHID</td>
<td>Machine ID byte.</td>
</tr>
<tr>
<td>[00]</td>
<td>0...</td>
<td>II</td>
</tr>
<tr>
<td>[01]</td>
<td>0...</td>
<td>II+</td>
</tr>
<tr>
<td>[10]</td>
<td>0...</td>
<td>IIe</td>
</tr>
<tr>
<td>[11]</td>
<td>0...</td>
<td>III emulation</td>
</tr>
<tr>
<td>[00]</td>
<td>1...</td>
<td>Future expansion</td>
</tr>
<tr>
<td>[01]</td>
<td>1...</td>
<td>Future expansion</td>
</tr>
<tr>
<td>[10]</td>
<td>1...</td>
<td>IIC</td>
</tr>
<tr>
<td>[11]</td>
<td>1...</td>
<td>Future expansion</td>
</tr>
<tr>
<td>[00]</td>
<td>...</td>
<td>Unused</td>
</tr>
<tr>
<td>[01]</td>
<td>...</td>
<td>48K</td>
</tr>
<tr>
<td>[10]</td>
<td>...</td>
<td>64K</td>
</tr>
<tr>
<td>[11]</td>
<td>...</td>
<td>128K</td>
</tr>
<tr>
<td>[XX]</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>[0]</td>
<td>...</td>
<td>No 80-column card</td>
</tr>
<tr>
<td>[1]</td>
<td>...</td>
<td>80-column card present</td>
</tr>
<tr>
<td>[0]</td>
<td>...</td>
<td>No compatible clock</td>
</tr>
<tr>
<td>[1]</td>
<td>...</td>
<td>Compatible clock present</td>
</tr>
<tr>
<td>BF99</td>
<td>SLTBYT</td>
<td>Slot ROM map (bit on indicates ROM present).</td>
</tr>
<tr>
<td>BF9A</td>
<td>PFXPTR</td>
<td>Prefix flag (0 indicates no active prefix).</td>
</tr>
<tr>
<td>BF9B</td>
<td>MLACTIVE</td>
<td>MLI active flag (1... indicates active).</td>
</tr>
<tr>
<td>BF9C-BF9D</td>
<td>CMDADDR</td>
<td>Last MLI call return address.</td>
</tr>
<tr>
<td>BF9E</td>
<td>SAVERX</td>
<td>X-register save area for MLI calls.</td>
</tr>
<tr>
<td>BF9F</td>
<td>SAVEX</td>
<td>Y-register save area for MLI calls.</td>
</tr>
</tbody>
</table>

### Language Card Bank Switching Routines

<table>
<thead>
<tr>
<th>ADDR</th>
<th>LABEL</th>
<th>CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>BF9A</td>
<td>EXIT</td>
<td>Language card entry and exit routines.</td>
</tr>
<tr>
<td>BF9B</td>
<td>EXIT1</td>
<td></td>
</tr>
<tr>
<td>BF9C</td>
<td>EXIT2</td>
<td></td>
</tr>
<tr>
<td>BF9D</td>
<td>MLIENTRY</td>
<td></td>
</tr>
</tbody>
</table>

### Interrupt Routines

- **BF90-BF93**
  - IRQ1
  - IRQ1
- **BF94-BF97**
  - IRQ4
  - IRQ5
  - IRQ6
  - IRQ7
  - IRQ8

### Data

- **BF94**
  - BNKBYTE1
- **BF95**
  - BNKBYTE2
- **BF96-BF98**
  - Switch on language card and call system death handler ($D1E4). |

### Version Information

- **BF9A**
  - IBKVER
- **BF9B**
  - MLIVER
- **BF9C**
  - MBKVER
- **BF9D**
  - KBKVER
- **BF9E**
  - KBASE
- **BF9F**
  - KBVER

### Notes

- **BF99**
  - Slot ROM map (bit on indicates ROM present).
- **BF9A**
  - Prefix flag (0 indicates no active prefix).
- **BF9B**
  - MLI active flag (1... indicates active).
- **BF9C-BF9D**
  - Last MLI call return address.
- **BF9E**
  - X-register save area for MLI calls.
- **BF9F**
  - Y-register save area for MLI calls.
Beneath Apple ProDOS Supplement

ProDOS Quit Code -- V1.0.1

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>MODULE STARTING ADDR</td>
</tr>
<tr>
<td>0024</td>
<td>Cursor Horizontal</td>
</tr>
<tr>
<td>0025</td>
<td>Cursor Vertical</td>
</tr>
</tbody>
</table>

1000 ****** ZERO PAGE

0280 Prefix Buffer
1800 Buffer
BF00 MLI Entry
BF50 Bitmap

1000 ****** SOFT SWITCH

C000 Keyboard
C002 Disable 80 column
C004 Disable 80 column
C00F Select alternate c
C010 Keyboard Strobe
C082 ROM select

1000 ****** MONITOR EQUIPMENT

FC5A Home
FC9C Clear to end of line
FD0C Read a key
FD0E Output a Carriage
FD2D Output a Character
FE99 Set Video
FF93 Set Keyboard
F03A Sound Bell

63

ProDOS Quit Code -- V1.0.1 -- 1 JAN 84

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>INITIALIZE **********</td>
</tr>
<tr>
<td>1000</td>
<td>Select ROM (C082)</td>
</tr>
<tr>
<td>1003</td>
<td>Set Video (FE93)</td>
</tr>
<tr>
<td>1006</td>
<td>Set Keyboard (FF93)</td>
</tr>
<tr>
<td>1009</td>
<td>Disable 80 column card (C00C)</td>
</tr>
<tr>
<td>100C</td>
<td>Select Alternate character set (C00F)</td>
</tr>
<tr>
<td>100F</td>
<td>Disable 80 column store (C008)</td>
</tr>
</tbody>
</table>

1012 INITIALIZE MEMORY BITMAP

- 1 -- 1 JAN 84 -
ProDOS Quit Code -- V1.0.1 -- 1 JAN 84

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1083</td>
<td>Decrement cursor horizontal position</td>
</tr>
<tr>
<td>1083</td>
<td>Decrement counter</td>
</tr>
<tr>
<td>1086</td>
<td>Clear to end of line &lt;FC9C&gt;</td>
</tr>
<tr>
<td>1089</td>
<td>Try again &gt;&gt;1063</td>
</tr>
<tr>
<td>108C</td>
<td>Continue if greater than or equal to BACKSPACE &gt;&gt;1094</td>
</tr>
<tr>
<td>108E</td>
<td>Else, sound Bell &lt;FP3A&gt;</td>
</tr>
<tr>
<td>1091</td>
<td>Try again &gt;&gt;1063</td>
</tr>
<tr>
<td>1094</td>
<td>Is it less than or equal to &quot;z&quot;?</td>
</tr>
<tr>
<td>1096</td>
<td>Yes, keep checking &gt;&gt;109A</td>
</tr>
<tr>
<td>1098</td>
<td>Turn off lowercase</td>
</tr>
<tr>
<td>109A</td>
<td>Is it less than &quot;.&quot;?</td>
</tr>
<tr>
<td>109C</td>
<td>Yes, Invalid - try again &gt;&gt;108E</td>
</tr>
<tr>
<td>109E</td>
<td>Is it greater than &quot;z&quot;?</td>
</tr>
<tr>
<td>10A0</td>
<td>Yes, Invalid - try again &gt;&gt;108E</td>
</tr>
<tr>
<td>10A2</td>
<td>Is it less than or equal to &quot;9&quot;?</td>
</tr>
<tr>
<td>10A4</td>
<td>Yes, keep checking &gt;&gt;10A4</td>
</tr>
<tr>
<td>10A6</td>
<td>Is it less than &quot;A&quot;?</td>
</tr>
<tr>
<td>10A8</td>
<td>Yes, Invalid - try again &gt;&gt;108E</td>
</tr>
<tr>
<td>10AA</td>
<td>Else, valid character - increment counter</td>
</tr>
<tr>
<td>10AB</td>
<td>Found 39 characters?</td>
</tr>
<tr>
<td>10AD</td>
<td>Yes, then start all over &gt;&gt;1075</td>
</tr>
<tr>
<td>10AF</td>
<td>Put valid character in buffer (0280)</td>
</tr>
<tr>
<td>10B2</td>
<td>and Print it &lt;FDED&gt;</td>
</tr>
<tr>
<td>10B5</td>
<td>Go back to print buffer for more &gt;&gt;1063</td>
</tr>
<tr>
<td>10B8</td>
<td>Check counter</td>
</tr>
<tr>
<td>10BA</td>
<td>If 0 then go on &gt;&gt;10CE</td>
</tr>
<tr>
<td>10BC</td>
<td>Else, save length (0280)</td>
</tr>
<tr>
<td>10BF</td>
<td>Call MLI (SET PREFIX) &lt;BF00&gt;</td>
</tr>
<tr>
<td>10C2</td>
<td>Data: SET PREFIX command number</td>
</tr>
<tr>
<td>10C3</td>
<td>Data: Pointer to Parameter list</td>
</tr>
<tr>
<td>10C5</td>
<td>Carry on if no error &gt;&gt;10CE</td>
</tr>
<tr>
<td>10C7</td>
<td>Sound Bell &lt;FP3A&gt;</td>
</tr>
<tr>
<td>10CA</td>
<td>Force branch to</td>
</tr>
<tr>
<td>10CC</td>
<td>always be taken &gt;&gt;1075</td>
</tr>
</tbody>
</table>

10CE ****** GET APPLICATION NAME ****************************

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>10CE</td>
<td>Clear Screen and Home cursor &lt;FC58&gt;</td>
</tr>
<tr>
<td>10D1</td>
<td>Go down 1 line &lt;FD8E&gt;</td>
</tr>
<tr>
<td>10D4</td>
<td>Get Pointer to Prompt2 (Application)</td>
</tr>
<tr>
<td>10D6</td>
<td>and store it in Print Routine (11E9)</td>
</tr>
<tr>
<td>10DE</td>
<td>Print it &lt;11E6&gt;</td>
</tr>
<tr>
<td>10E1</td>
<td>Position to line 3</td>
</tr>
<tr>
<td>10E8</td>
<td>Initialize counter</td>
</tr>
<tr>
<td>10EA</td>
<td>Output a RUB</td>
</tr>
<tr>
<td>10F1</td>
<td>Poll Keyboard latch (C000)</td>
</tr>
<tr>
<td>10F4</td>
<td>Loop until keypress found &gt;&gt;10F1</td>
</tr>
<tr>
<td>10F6</td>
<td>Clear latch (C010)</td>
</tr>
</tbody>
</table>

10F9 ****** LOAD AND EXECUTE APPLICATION ******

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>10F9</td>
<td>is it ESCAPE?</td>
</tr>
<tr>
<td>10FB</td>
<td>No, keep checking &gt;&gt;103</td>
</tr>
<tr>
<td>10FD</td>
<td>Yes, get Cursor horizontal position</td>
</tr>
<tr>
<td>10FF</td>
<td>If not 0 try again &gt;&gt;10CE</td>
</tr>
<tr>
<td>1101</td>
<td>If 0 start all over again &gt;&gt;10CC</td>
</tr>
<tr>
<td>1103</td>
<td>Is it CANCEL?</td>
</tr>
<tr>
<td>1105</td>
<td>Yes, try again &gt;&gt;10CE</td>
</tr>
<tr>
<td>1107</td>
<td>Is it TAB?</td>
</tr>
<tr>
<td>1109</td>
<td>Yes, sound Bell - try again &gt;&gt;1114</td>
</tr>
<tr>
<td>110B</td>
<td>Is it BACKSPACE?</td>
</tr>
<tr>
<td>110D</td>
<td>No, keep checking &gt;&gt;1112</td>
</tr>
<tr>
<td>110F</td>
<td>Yes, then handle it &gt;&gt;1105</td>
</tr>
<tr>
<td>1112</td>
<td>Continue if greater than or equal to BACKSPACE &gt;&gt;111A</td>
</tr>
<tr>
<td>1114</td>
<td>Sound Bell &lt;FP3A&gt;</td>
</tr>
<tr>
<td>1117</td>
<td>Go back and try again &gt;&gt;10EA</td>
</tr>
<tr>
<td>111A</td>
<td>Is it CARRIAGE RETURN?</td>
</tr>
<tr>
<td>111C</td>
<td>Yes, then go load Application &gt;&gt;1147</td>
</tr>
<tr>
<td>111E</td>
<td>Is it less than or equal to &quot;g&quot;?</td>
</tr>
<tr>
<td>1120</td>
<td>Yes, keep checking &gt;&gt;1124</td>
</tr>
<tr>
<td>1122</td>
<td>turn off lower case</td>
</tr>
<tr>
<td>1124</td>
<td>Is it less than &quot;.&quot;?</td>
</tr>
<tr>
<td>1126</td>
<td>Yes, Invalid - try again &gt;&gt;1114</td>
</tr>
<tr>
<td>1128</td>
<td>Is it greater than &quot;z&quot;?</td>
</tr>
<tr>
<td>112A</td>
<td>Yes, Invalid - try again &gt;&gt;1114</td>
</tr>
<tr>
<td>112C</td>
<td>Is it less than or equal to &quot;g&quot;?</td>
</tr>
<tr>
<td>112E</td>
<td>Yes, keep checking &gt;&gt;1134</td>
</tr>
<tr>
<td>1130</td>
<td>Is it less than &quot;A&quot;?</td>
</tr>
<tr>
<td>1132</td>
<td>Yes, Invalid - try again &gt;&gt;1114</td>
</tr>
<tr>
<td>1134</td>
<td>Else, valid character - save it</td>
</tr>
<tr>
<td>1135</td>
<td>Clear to end of line &lt;FC9C&gt;</td>
</tr>
<tr>
<td>1138</td>
<td>Retrieve character</td>
</tr>
<tr>
<td>1139</td>
<td>And Print it &lt;FDED&gt;</td>
</tr>
<tr>
<td>113C</td>
<td>Increment counter</td>
</tr>
<tr>
<td>113D</td>
<td>Found 39 characters?</td>
</tr>
<tr>
<td>113F</td>
<td>Yes, start again &gt;&gt;1105</td>
</tr>
<tr>
<td>1141</td>
<td>No, save character in buffer (0280)</td>
</tr>
<tr>
<td>1144</td>
<td>and go get another &gt;&gt;10EA</td>
</tr>
</tbody>
</table>

1147 ****** Output a blank ******

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1147</td>
<td>Output a blank</td>
</tr>
<tr>
<td>114C</td>
<td>Call MLI (GET FILE INFO) &lt;BF00&gt;</td>
</tr>
<tr>
<td>1152</td>
<td>Data: GET_FILE_INFO command number</td>
</tr>
<tr>
<td>1153</td>
<td>Data: Pointer to Parameter list</td>
</tr>
<tr>
<td>1155</td>
<td>Continue if no error &gt;&gt;115A</td>
</tr>
<tr>
<td>1157</td>
<td>Else, go to Error Handler &gt;&gt;11F6</td>
</tr>
</tbody>
</table>
ProDOS Quit Code -- V1.0.1

ADDR DESCRIPTION/CONTENTS

11CA READ good?
11CB go to Error Handler >>11C7
11CD yes, execute application >>2000

11D0 BACKSPACE ROUTINE

11D2 exit routine >>11E3
11D4 decrement counter
11D5 put a space
11D7 put a space <FDED>
11D9 get cursor back 1 space
11D3 return to get another character >>10EA

11E6 PRINT TEXT ROUTINE

11E6 initialize offset
11E8 character (11E8)
11EB if it is 0 then exit >>11F5
11F2 put it <FDED>
11F3 10 more offset
11F5 get another character unless we've done 256 >>11E8
11F9 return to caller

11F6 PRINT ERROR MESSAGE

11F6 Accumulator (Error Number)
11F9 position to line 12
11F7 Error number
1201 if it $01?
1203 then keep checking >>1211
1205 get Pointer to Error1 (Not System file)
1207 up store it in Print Routine (11E9)
120F which always taken >>1237
1211 if it $40?
1213 $40, then indicate Error3 >>122D
1215 it $44?
1217 it $44?
1219 it $44?
121B $44, then indicate Error3 >>122D
121D it $46?
121F it $46?
1221 $46, then indicate Error3 >>122D
1223 store it in Print Routine (11E9)
1229 which always taken >>1237
122D up store it in Print Routine (11E9)
1237 Writ Error message <11E6>
123A position to line 0

Beneath Apple ProDOS Supplement
ProDOS Quit Code -- V1.0.1 -- 1 JAN 84  NEXT OBJECT ADDR: 123E

123E  Return to Get Application code >>10D1

1241  ********** ASCII TEXT ************************************

  'Prompt1
1241  'ENTER  PREFIX (PRESS "RETURN" TO ACCEPT)'

  'Prompt2
1269  'ENTER  PATHNAME OF NEXT APPLICATION'

  Error1
128C  Ring Bell
128D  'NOT  A TYPE "SYS" FILE'

  Error2
12A3  Ring Bell
12A4  'I/O  ERROR   '

  Error3
12BA  Ring Bell
12BB  'FILE/PATH  NOT FOUND   '

12D1  ********** PARAMETER LISTS ***********************

  GET_FILE_INFO Parmlist
12D1  ParmCount
12D2  Pathname
12D4  Access
12D5  File Type
12D6  Aux Type
12D8  Storage Type
12D9  Blocks Used
12DB  Datetime (modified)
12DF  Datetime (creation)

  OPEN Parmlist
12E3  Parmcount
12E4  Pathname
12E6  I/O Buffer
12E8  Reference Number

  CLOSE Parmlist
12E9  Parmcount
12EA  Reference Number

ProDOS Quit Code -- V1.0.1 -- 1 JAN 84  NEXT OBJECT ADDR: 12EA

12EB  Parmcount
12EC  Reference Number
12ED  Data Buffer
12EF  Request Count
12F1  Transfer Count

  GET_EOF Parmlist
12F3  Parmcount
12F4  Reference Number
12F5  EOF Mark

  GET_SET_PREFIX Parmlist
12FB  Parmcount
12F9  Pathname

    ******** UNUSED **************
12FB  --- >>0005
Disk II Device Driver -- V1.0.1 -- 1 JAN 84  NEXT OBJECT ADDR: F800

MODULE STARTING ADDRESS

**-zero page equates**

003A Checksum
003A Workbyte
003E Slot (Temporary)
0042 Command
0043 Unit Number
0044 I/O Buffer Pointer (low)
0045 I/O Buffer Pointer (high)
0046 Block Number (low)
0047 Block Number (high)

**internal equates**

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

**external equates**

C080 Phase Zero Off
C088 Motor Off
C089 Motor On
C08A Drive Select
C08C Read Data Register
C08D Write Data Register
C08E Set Read Mode
C08F Set Write Mode
C09C Read Data Register (slot 6)

**convert block to track/sector**

F800 Get Block Number
F804 Is Block Number good? (FB56)
F807 Yes, if less than $100 >>F810
F80A No, if greater than or equal to $200 >>F836
F80E No, if greater than or equal to $118 >>F836
F810 Convert Block Number to a Track and Sector
F812 ---
F816 00000T TTTTABC
F817 . >>F812
F819 .
F81A . >>F81E
F81C 0TTTTT 000BC0A
F81E ---
F822 Preserve Sector Number
F823 Execute command <F83A>
F826 Restore Sector Number -- Was prior action ok?
F827 No, then exit >>F835
F829 Increment Buffer Pointer
F82B Increment Sector Number by 2 for rest of Block
F82D Execute command <F83A>
F830 Increment Buffer Pointer (to start of block)
F832 Get error number (if any - 0 indicates no error) (FB58)
F835 Return to caller

F836 ******** I/O ERROR ROUTINE ***********************

F836 Indicate "I/O Error"
F838 Set Carry flag
F839 Return to caller

**main routine**

F83A Set recalibration count to 1
F83F Preserve sector number (FB57)
F842 Get "Unitnum" DSSS0000
F844 Strip out Drive 00000000
F846 Preserve slot number
F848 Check for slot change, turn off motor if so <F998>
F84B See if motor is on <FCDA>
F84E Save test results
F851 Initialize counter for delay routine (FB70)
F856 See if slot or drive has changed (FB59)
F859 Update "current" unit number (FB59)
F85C Save test results
F85D Put drive number in Carry flag
F85E Turn motor on (C089)
F864 Select appropriate drive (C08A)
F867 Check test results - Same slot/drive?
F868 Yes, then skip delay >>F874
F86B Wait for new Drive
Disk II Device Driver -- V1.0.1 -- 1 JAN 84

NEXT OBJECT ADDR: F86D

F86D Preserve error number (FB58)
F86F Turn motor off (C088)
F873 Return to caller

F874 Write data - Good write? <FDB0>
F875 Yes, then exit >>F859
F879 Indicate "Write-protect error"
F87B Branch always taken >>F85C

F87C Get status
F87D Get Slot number
F87E Check "write-protect" status (C08E)
F87F Put result in Carry flag
F880 Select read mode (C08C)
F881 Exit with appropriate status >>F87F

F88C Locate desired track
F88D Double the track number for proper phase
F88F. Preserve destination track #2 (F86F)
F890 Turn all phases off >>F925
F891 Get offset into Device Track Table <FCF1>
F892 Get track (FB59)
F893 Update "current" track (FB5A)
F894 Get destination track (F86F)
F895 Update Device Track Table (FB59)
F896 Move arm to desired track <<F933>
F897 Initialize phase number, starting with 3
F898 ----
F899 Clear a phase <<F98A>
F89B Decrement phase number - More to do?
F89C Yes, then continue until all phases done >>F927
F89E Divide track number by 2 (FB5A)
F89F Return to caller

F8A3 ARM move routine
F8A4 Preserve track to find (FB72)
F8A5 Are we already there? (FB5A)
F8A6 Yes, then set appropriate phase and exit >>F987
F8A7 Yes, then clear prior phase and exit >>F983
F8A9 Positive delta-tracks - go move arm out >>F955
F8AA Negative delta-tracks - Get absolute value delta-tracks less 1
F8AF Increment current phase to move in (FB5A)

F8AD to come up to speed <<FB85>
F8AD Is command a status request?
F8AE Yes, then do not move disk arm >>F873
F8AF Get track number for current request
F8B0 And go there <<F90C
F8B1 Check test results - Was motor on?
F8B2 Yes, then skip delay >>F890
F8B3 Wait for Drive
F8B4 to come up to speed <<FB85>

F8B5 ---
F8B6 --
F8B7 --
F8B8 --
F8B9 --
F8BA --
F8BB --
F8BC --
F8BD --
F8BE --
F8BF --
F8C0 --
F8C1 --
F8C2 --
F8C3 --
F8C4 --
F8C5 --
F8C6 --
F8C7 --
F8C8 --
F8C9 --
F8CA --
F8CB --
F8CC --
F8CD --
F8CE --
F8CF --
F8D0 --
F8D1 --
F8D2 --
F8D3 --
F8D4 --
F8D5 --
F8D6 --
F8D7 --
F8D8 --
F8D9 --
F8DA --
F8DB --
F8DC --
F8DD --
F8DE --
F8DF --
F8E0 --
F8E1 --
F8E2 --
F8E3 --
F8E4 --
F8E5 --
F8E6 --
F8E7 --
F8E8 --
F8E9 --
F8EA --
F8EB --
F8EC --
F8ED --
F8EE --
F8EF --
F8F0 --
F8F1 --
F8F2 --
F8F3 --
F8F4 --
F8F5 --
F8F6 --
F8F7 --
F8F8 --
F8F9 --
F8FA --
F8FB --
F8FC --
F8FD --
F8FE --
F8FF --
Disk II Device Driver -- V1.0.1 -- 1 JAN 84 NEXT OBJECT ADDR: F953

ADDR DESCRIPTION/CONTENTS

F953 Branch always taken >>F95A
F955 Compute absolute value delta
F957 Decrement current phase to tracks less 1
F95A Compair delta-tracks with phove out (F95A)
F95D Use smaller value for offsettases moved (F968)
F962 Are we pointing at last table to xelay tables >>F962
F964 Yes, then continue to use cve value yet?
F966 Else, use new offset
F967 Set Carry flag for set phase
F968 Set a phase <F987>
F96B Get delay value from table ||
F96E Delay <FB85> (F673)
F971 Get prior phase number (FB71)
F974 Clear Carry flag for clear ||
F975 Clear a phase <F98A> phase operation
F978 Get delay value from table ||
F97B Delay <FB85> (F67C)
F97E Increment phases moved (FB6N)
F983 Delay <FB85>
F987 Get "current" phase number ||
F98A Use low two bits only, zero (F95A)
F98C Multiply by two and bring into one three - 00000000
F98D Merge in slot number
F98F Put in X-reg for following ||
F990 Toggle appropriate phase (C)operation
F993 Restore slot number to X-reg (88)
F995 Return to caller ||

F996 ******** TABLE 1 ********************

Read Translate Table with..............................
Bit mask Tables and Epilog
Read Translate

Bit mask Tables

FA00 00000000
FA01 00000000
FA02 10000000
FA03 10000000

Read Translate

Bit mask 2

FA0C 00000000
FA0D 01000000
FA0E 01000000
FA0F 01000000

Read Translate

Epilog Table ($DE,$AA,$EB)
Read Translate

Bit Mask 3

FA00 00000000
FA01 00001000
FA02 00001000
FA03 00001000

Read Translate

FA00 ******** TABLE 2 ******************************************************

Write Translate Table
Every 4th byte starting at $FA03

FA00 Entry for Bit Mask 1
FA01 Entry for Bit Mask 2
FA02 Entry for Bit Mask 3
FA03 Entry for Write Translate

FA00 ******** AUXILIARY BUFFER ******************************************

FB00 Auxiliary Buffer ($56 bytes) >>0056

FB56 ******** VARIABLE AREA ***************************************

FB56 Track number
FB57 Sector number
FB58 Error number

Disk Device Track Table

FB59 Table Entry
FB59 Current Unit
FB5A Current Track
FB5B Slot 1, Devices 1 & 2
FB5C Slot 2, Devices 1 & 2
FB5D Slot 3, Devices 1 & 2
FB5E Slot 4, Devices 1 & 2
FB5F Slot 5, Devices 1 & 2
FB60 Slot 6, Devices 1 & 2
FB61 Slot 7, Devices 1 & 2
FD0F into zero page for timing
FD11 Use SPF for "sync" byte
FD13 Write first "sync" byte (C66F)
FD19 Set counter for four more
FD1C Delay so that writes occur
FD1D Exactly on 40 cycle loops
FD1E ---

Beneath Apple Pro...

Disk II Device Driver -- V1.0.1 -- 1 JAN 84
NEXT OBJECT ADDR: FCCA

FCCA Yes, then continue with carry clear >>FCCD
FCCD Get carry flag indicating error
FCD0 Get byte we stored away, we have time now
FCD2 Set proper offset
FCD9 Store byte in Primary buffer (offset $55)
FCD9 Return to caller

FCD3 ************** UPDATE DEVICE TRACK TABLE **************
FCD3 Get offset into Device Track Table (FCD1)
FCD6 Update Device Track Table (FB59)
FCD9 Return to caller

FCD4A ************** DETERMINE IF DRIVE IS ON (DATA CHANGING)**************
FCD4A Get slot number
FCD5C Initialize counter
FCD6 Read data register (C68C)
FCD60 Delay 25 cycles <FCFO
FCD65 Has data register changed? (C68C)
FCD9 Yes, then exit >>FCFO
FCD9 Just in case indicate No Device Connected Error
FCD90 Decrement count - 256 tries yet?
FCD9E No, try again >>FCD9E
FCD9 Return to caller

FCD1 ************** CONVERT SLOT/DRIVE TO TABLE OFFSET * SKP 1***
FCF1 Preserve A-register
FCF2 Get unit number DSSSSSSS
FCF4 Divide by 16 00000000 D
FCF8 Put Drive into Carry 00000000 D
FCFA Strip out Drive 00000000 D
FCFC Roll left 00000000 D
FCFD Put result in X-register
FCFE Restore A-register
FCFF Return to caller

FD00 ************** WRITE DATA ROUTINE *****************
FD00 Set Carry flag (anticipate error)
FD04 Is diskette "write-protected" (C68E)
FD07 No, then continue on >>FD0C
FD09 Go to error routine >>FD0D
FD0C Put transition byte from secondary buffer (FB60)
Beneath Apple...

Disk II Device Driver -- V1.0.1 -- 1 JAN 84  NEXT OBJECT ADDR: FD9E

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>FD9E</td>
<td>Lookup &quot;disk byte&quot; in table (FA83)</td>
</tr>
<tr>
<td>FD9F</td>
<td>Get slot</td>
</tr>
<tr>
<td>FD9A</td>
<td>Write &quot;disk byte&quot; (C08D)</td>
</tr>
<tr>
<td>FD9B</td>
<td>Get data byte (Primary buffer - page 2) (1100)</td>
</tr>
<tr>
<td>FD9C</td>
<td>Increment offset - Done yet?</td>
</tr>
<tr>
<td>FD9D</td>
<td>No, then do another &gt;&gt;FD9B</td>
</tr>
<tr>
<td>FD9E</td>
<td>Yes, then go write checksum &gt;&gt;FD9B</td>
</tr>
<tr>
<td>FD9F</td>
<td>--- &gt;&gt;FD9C</td>
</tr>
<tr>
<td>FD9A</td>
<td>Get last byte (0038)</td>
</tr>
<tr>
<td>FD9B</td>
<td>Write it (C08D)</td>
</tr>
<tr>
<td>FD9C</td>
<td>Delay 14 cycles for correct timing</td>
</tr>
<tr>
<td>FD9D</td>
<td>Use last byte in Primary buffer as checksum</td>
</tr>
<tr>
<td>FD9E</td>
<td>Lookup &quot;disk byte&quot; (FA83)</td>
</tr>
<tr>
<td>FD9F</td>
<td>Get slot</td>
</tr>
<tr>
<td>FD9A</td>
<td>Write &quot;disk byte&quot; (C08D)</td>
</tr>
<tr>
<td>FD9B</td>
<td>Initialize offset into &quot;epilog&quot; table</td>
</tr>
<tr>
<td>FD9C</td>
<td>Delay 11 cycles for correct timing</td>
</tr>
<tr>
<td>FD9D</td>
<td>Load &quot;epilog&quot; from table ($DE, $AA, $EB, $FF) (FC4)</td>
</tr>
<tr>
<td>FD9E</td>
<td>Go write it &lt;FDE9&gt;</td>
</tr>
<tr>
<td>FD9F</td>
<td>Write &quot;disk byte&quot; (C08D)</td>
</tr>
<tr>
<td>FD9A</td>
<td>Initialize offset into &quot;epilog&quot; table</td>
</tr>
<tr>
<td>FD9B</td>
<td>Delay 11 cycles for correct timing</td>
</tr>
<tr>
<td>FD9D</td>
<td>Load &quot;epilog&quot; from table ($DE, $AA, $EB, $FF) (FC4)</td>
</tr>
<tr>
<td>FD9E</td>
<td>Go write it &lt;FDE9&gt;</td>
</tr>
<tr>
<td>FD9F</td>
<td>Increment offset</td>
</tr>
<tr>
<td>FD9A</td>
<td>Done all four yet?</td>
</tr>
<tr>
<td>FD9B</td>
<td>No, then do another &gt;&gt;FD9D</td>
</tr>
<tr>
<td>FD9C</td>
<td>Clear Carry flag (no error)</td>
</tr>
<tr>
<td>FD9D</td>
<td>Select read mode (C08E)</td>
</tr>
<tr>
<td>FD9E</td>
<td>Return to caller</td>
</tr>
</tbody>
</table>

FDE9 **************************** WRITE A BYTE SUBROUTINE ****************************

FDE9 | Wait 9 cycles before write |
| FDEA | Wait 7 cycles before write |
| FDEB | Put A-register in data register (C08D) |
| FDEC | And write data register (C08C) |
| FDEF | Return to caller |

FDF0 **************************** PREPENLIZE BLOCK ROUTINE ****************************

FDF0 | Get buffer pointer |
<p>| FDF1 | Add 82 to buffer address |
| FDF2 | To access top third of buffer &gt;&gt;FDFE |
| FDF3 | Store result in code below (FE30) |
| FDF4 | Subtract $54 from buffer address |
| FDF5 | To access middle third of buffer &gt;&gt;FDF6 |
| FDF6 | Store result in code below (FE25) |
| FDF7 | Subtract $AA from buffer address |
| FDF8 | To access bottom third of buffer &gt;&gt;FDF9 |
| FDF9 | Store result in code below (FE1B) |
| FDF0 | Initialize offset |
| FDF1 | Get data byte (bottom third) | XXXXXXXX (1000) |
| FDF2 | Get last two bits 000000AB |
| FDF3 | Put in X-reg for table lookup |
| FDF4 | Use lookup to reposition bits 0000BA00 (F9E0) |</p>
<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>FE23</td>
<td>Save result on stack</td>
</tr>
<tr>
<td>FE24</td>
<td>Get data byte (middle third)</td>
</tr>
<tr>
<td>FE27</td>
<td>Get last two bits</td>
</tr>
<tr>
<td>FE29</td>
<td>Put in X-reg for table lookup</td>
</tr>
<tr>
<td>FE2A</td>
<td>Get current value from stack</td>
</tr>
<tr>
<td>FE2B</td>
<td>Merge new bits using table</td>
</tr>
<tr>
<td>FE2E</td>
<td>Save result on stack</td>
</tr>
<tr>
<td>FE2F</td>
<td>Get data byte (top third)</td>
</tr>
<tr>
<td>FE32</td>
<td>Get last two bits</td>
</tr>
<tr>
<td>FE34</td>
<td>Put in X-reg for table lookup</td>
</tr>
<tr>
<td>FE35</td>
<td>Get current value from stack</td>
</tr>
<tr>
<td>FE36</td>
<td>Merge new bits using table</td>
</tr>
<tr>
<td>FE39</td>
<td>Save result on stack</td>
</tr>
<tr>
<td>FE3A</td>
<td>Get offset into primary buffer</td>
</tr>
<tr>
<td>FE3B</td>
<td>Compute offset into Auxiliary buffer</td>
</tr>
<tr>
<td>FE3D</td>
<td>Put in X-reg</td>
</tr>
<tr>
<td>FE3E</td>
<td>Get data byte just created</td>
</tr>
<tr>
<td>FE3F</td>
<td>Store it in Auxiliary buffer (FB80)</td>
</tr>
</tbody>
</table>
| FE42 | Increment offset primary buffer, done yet?
| FE43 | No, then do another >>FE1A |
| FE45 | Get low order byte of buffer |
| FE47 | Subtract 1 (offset to last byte in buffer) |
| FE49 | Save it for later |
| FE4A | Get low order byte of buffer |
| FE4C | Modify code in Write Data Routine (offset) (FB52) |
| FE4F | Buffer on page boundary? Yes, skip ahead >>FE5F |
| FE51 | Else, compute offset to last byte |
| FE53 | Before page boundary |
| FE54 | Get byte (page boundary -1) |
| FE56 | Point at next byte (page boundary) |
| FE57 | Exclusive-or them together |
| FE59 | Strip off last two bits |
| FE5B | Put in X-reg for table lookup |
| FE5C | Get "disk byte" from table (transition byte) (FA03) |
| FE5F | Save result (0 indicates page boundary) |
| FE61 | Buffer on page boundary? Yes skip ahead >>FE6F |
| FE63 | Get offset to last byte in buffer |
| FE65 | Carry indicates odd or even buffer start |
| FE66 | Get byte (page boundary) |
| FE68 | Did buffer start on odd byte? Yes skip >>FE6D |
| FE6A | Point at next byte (page boundary +1) |
| FE6B | Exclusive-or them together |
| FE6D | Save result |
| FE6F | Point at last byte in buffer |
| FE71 | Get last byte in buffer |
| FE73 | Strip off last two bits |
| FE75 | Save result ("checksum byte") |
| FE77 | Get high order byte of buffer |
| FE79 | Modify code in Write Data Routine (FB55) |
| FE8C | Get 16-bit number for this operation |
| FE8E | Modify code in Write Data Routine (FB5D) |

---

Disk II Device Driver -- V1.0.1 -- 1 JAN 84

**FE9A**

Return to caller

**FE9B**

************ DETERMINE IF SLOT/DRIVE HAS CHANGED ***************

FE9B Compare unit number with "current" unit number (FB59)

FE9E Put "current" drive in Carry

FE9F Has slot changed? No, then exit >>FEBD

FE9A Get "current" slot

FEAB Put in X-register

FEAC Exit if Slot 0 >>FEBD

FEAE Is "current" motor on? <FDCC>

FEBl No, then exit >>FEBD

FEBB Wait until "current" motor is off (FB70)

FEBB Or else timeout >>FEA6

FEBD Return to caller

FEBE Unused >>FE42

003B
### IRQ Handler -- V1.0.1 -- 1 JAN 84

<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>&quot;&quot;</td>
</tr>
<tr>
<td></td>
<td>* Interrupt handler:</td>
</tr>
<tr>
<td></td>
<td>* Resides at SFF9B</td>
</tr>
<tr>
<td></td>
<td>*</td>
</tr>
<tr>
<td></td>
<td>*</td>
</tr>
<tr>
<td></td>
<td>*</td>
</tr>
</tbody>
</table>
|      | * Versions 1.0.1 -- 1 JAN 84 *
|      | **GLOBAL PAGE EQUATES ************
| BF56 | Temporary storage 1 |
| BF57 | Temporary storage 2 |
| BF8B | A register save area |
| BF8D | Bank ID byte |
| BF93 | IRQ exit code |
| FF98 | **EXTERNAL EQUATES ************ |
| D900 | RAM/ROM test byte |
| C982 | ROM Select |
| C98B | Bank1 Select |
| FF9B | **IRQ CODE ** |
| FF9B | Put A-Register on stack |
| FF9C | Get Accumulator value from $45 |
| FF9E | and save it (BF56) |
| FF9F | Replace $45 with A-Register |
| FFAD | since it may have been destroyed |
| FF9B | Load Status register |
| FF95 | Restore onto stack |
| FF96 | Isolate B flag - Was it a BRK? |
| FF95 | Yes, skip Interrupt stuff >>F9C |
| FF9A | Else, Check location D900 (D900) |
| FFAD | Do we have RAM active |
| FF9F | Yes, indicate so >>FFB3 |
| FF91 | Else, indicate ROM |
| FF9B | Update Bank ID byte (BF8D) |
| FF96 | Also save temporarily (BF57) |
| FF99 | Push ($BBF50) address of |
| FF99 | routine to bank in Ram and |
| FF9B | call IRQ on the stack |
| FF9C | Push a new P-Register on stack with |
| FF9C | the Interrupt Disable flag set |
| FF92 | Push ($FA41) address less 1 of |
| FF92 | Monitor IRQ on the stack |
| FF93 | Select ROM - execution continues in ROM (C982) |
2000  MODULE STARTING ADDRESS

******************************************************************************
*  *  PRODOS BASIC INTERPRETOR RELOCATOR  
*  *  LOADED AS THE FIRST TWO BLOCKS  
*  *  OF BASIC.SYSTEM AT $2000.  
*  *  THIS ROUTINE MOVES THE BASIC  
*  *  INTERPRETOR TO HIGH MEMORY.  
*  *  VERSION 1.0.1 -- 1 JAN 84  
*  *  
******************************************************************************

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

0000  FROM POINTER FOR COPY
0001  TO POINTER FOR COPY
0003  CSWL VECTOR
0036  KSWL VECTOR
006F  APPLESOF START OF STRING
0073  APPLESOF HIMEM
00F2  APPLESOF TRACE FLAG

********* EXTERNAL ADDRESSES **********

0200  PATHNAME BUFFER
0208  PREFIX BUFFER
0281  START OF PREFIX NAME
03D9  WARMSTART VECTOR
03D3  COLDSTART VECTOR
03F0  BRK HANDLER ADDRESS
03F1  RESET HANDLER ADDRESS
03F3  POWER-UP BYTE
03F5  APPLESOF & VECTOR
03F8  CTL-Y VECTOR

******** SCREEN LINE ADDRESSES ********

0400  FIRST SCREEN BUFFER LINE
0408  SCREEN BUFFER LINE
0628  SCREEN BUFFER LINE

2000  ********** BASIC INTERP RELOCATOR ENTRY **********

2000  $00 --> $2400
2004  $02 --> $9A00
200E  COPY 35 PAGES
2011  COPY INTERP TO HIGH MEMORY AT $9A00 <207B>
2016  PAGE FOLLOWING INTERP IMAGE IS...
2018  BASIC GLOBAL PAGE IMAGE
201A  COPY THAT TO $BE00 <207B>
201D  TURN 80 COLUMNS OFF (C00C)
2020  SET NORMAL CHARACTER ATTRIBUTE <FE84>
2023  INITIALIZE SCREEN/WINDOW <FB2F>
2026  CLEAR SCREEN/HOME CURSOR <FC58>
202D  SET BITMAP TO MARK LOWER 48K FREE (BF58)
2033  EXCEPT PAGES 0 AND 1 AND
2035  TEXT PAGES 4 THROUGH 7 (BF58)
203D  MARK $9000--$BFFFF IN USE ...
2048  EXCEPT FOR $BA00--$BDFFF ARE FREE
SET MY CSWL/KSWL FOR INTERP INIT (216D)
COPY ALL 4 BYTES >>205D
THEN GO TO BASIC COLDSTART >>8000
(WE WILL GET CONTROL AT 208B AGAIN)

********** ERROR EXIT **********
* T ADDR: 204D

--
PRINT "UNABLE TO EXECUTE BASIC SYSTEM" (216F)
ALLOW REBOOT IF RESET PRESSED (43F4)
GO TO SLEEP FOREVER >>2079

********** COPY PAGES (80/1->82/3) **********

--
COPY FROM 50/1 TO 82/3

** A PAGE AT A TIME >>207B
** COUNTPAGES
** RETURN

********** CSWL INTERCEPT / CONTINUE **********

"^" APPLESOFT PROMPT?
NO...DON'T PRINT WHATEVER IT IS >>208A
F YES, APPLESOFT DONE SETTING UP (BE0E)

** POINT CSWL TO STANDARD OUTPUT
** CHECK LAST DEVICE USED (BF38)
** SET ONLINE PARAMETER TO THIS (216F)
** DRIVE ONE OR TWO? >>2085
** STORE DEFAULT DRIVE (D) (BE3D)
** ISOLATE SLOT FROM DEVICE NO.
** AND STORE DEFAULT SLOT (S) (BE1C)
** GET SLOT BYTE SHOWING CARDS PRESENT (BB99)
** PICK OFF ITS BITS ONE BY ONE
** SET OUTVCS AND INVCs TO $C500 (BE16)
** FOR ALL SLOTS WITH ROMS IN THEM (BE20)

** SET HIMEM TO $8600
** IN VARIOUS PLACES
** GOT A DEFAULT PREFIX? (BF9A)
** NO >>2105
** YES, MLI: GET PREFIX <BF00>
** ERROR? >>2142
** BACKSCAN PREFIX FOR "/"'S (#280)
** AND COUNT THEM IN $21EE (21F7)

********** FINISH UP AND GO TO BI **********

--
COPY WARMSTART JMP TO 3PAGE (21B0)
AND COLDSTART (#3D3)
AND CTL-Y (#3F8)
POINT & VECTOR (21B7)
TO SBE53 (CMD SCANNER) (#3F5)
COPY BRK HANDLER JMP ALSO (21B3)
AND RESET JMP (#3F2)
SET POWER-UP BYTE ACCORDINGLY (#3F4)
SET APPLESOFT IN NON-TRACE MODE
AND GO TO INTERPRETER >>BE00

********** VECTOR ADDRESSES **********
BI Relocator -- V1.0.1 -- 1 JAN 84
NEXT OBJECT ADDR: 21B3

21B3 BREAK HANDLER ADDRESS FOR 3PAGE
21B5 RESET HANDLER IS BASIC INTERP
21B7 APPLESOFT & GOES TO BI CMD SCANNER >>BE03

21BA *********** FIRST KSWL INTERCEPT ***********

21BA SET KSWL TO CURRENT DEVICE HANDLER (BE20)
21C4 RETURN LENGTH OF FIRST COMMAND (21E5)
21CB FOLLOWED BY A RETURN
21CA RETURN

21CB *********** DATA ***********

21CB CSWL (268B) INTERCEPT ADDR
21CD KSWL (21B1) INTERCEPT ADDR

21CF GET FILE INFO PARM LIST
21D0 FILE NAME IS AT $21DC

21E2 SET PREFIX PARM LIST
21E3 FOR PREFIX AT $21E4

21E5 STARTUP FILE NAME LENGTH ($7)
21E6 'STARTUP'

21ED NULL PREFIX
21EE "/"

21EF SAVED LENGTH OF STARTUP FILE NAME

21F0 ONLINE PARM LIST
21F2 PUT VOLUME NAME AT $281

21F4 SET PREFIX PARM LIST
21F5 PREFIX IS AT $280

21F7 NUMBER OF SUBLEVELS IN PREFIX +1

21F8 """""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84 NEXT OBJECT ADDR: 9A00

9A00 MODULE

**STARTING ADDRESS**

**PROCEDURES**

**BASIC INTERPRETER (BI)**

- This code starts in the third
- Lock of the file basic.system.
- It performs command handling
- For all built-in prods command and supports basic's file handling.

**VERSION**

- V1.0.1 -- 1 JAN 84

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

- 0024 CURSOR HOME
- 0028 SCREEN LINE CONTAL
- 0029 BASE ADDR
- 0033 MONITOR POS
- 0035 CRT DISPLAY REPT CHARACTER
- 0037 CRT VECTOR (CSWL)
- 0038 KEYBOARD 30
- 0039 SCRATCH COPY
- 003A SCRATCH COPY
- 003B SCRATCH COPY
- 003C SCRATCH COPY
- 003D SCRATCH COPY
- 003E SCRATCH COPY
- 003F SCRATCH COPY
- 0050 APPLESOFT: APPLESOFT VARIABLES
- 0051 LINE NUMBER
- 0067 APPLESOFT: APPLESOFT VARIABLES
- 0068 APPLESOFT: START OF PROGRAM PTR
- 0069 APPLESOFT: LOMEM (START OF VARS)
- 006A APPLESOFT: START OF ARRAY VARS PTR
- 006B APPLESOFT: START OF FREE ARE PTR
- 006C APPLESOFT: START OF STRINGS PTR
- 006D APPLESOFT: HIMEM (END OF STRINGS)
- 006E APPLESOFT: CURRENT LINE BEING EXECUTED
- 006F APPLESOFT: ADDR OF LINE AFTER FINDLINE
- 0070 APPLESOFT: END OF PROGRAM PTR
- 0071 APPLESOFT: END OF PROGRAM PTR
- 0072 APPLESOFT: END OF PROGRAM PTR
- 0073 APPLESOFT: END OF PROGRAM PTR
- 0074 APPLESOFT: END OF PROGRAM PTR
- 0075 APPLESOFT: END OF PROGRAM PTR
- 0076 APPLESOFT: END OF PROGRAM PTR
- 0077 APPLESOFT: END OF PROGRAM PTR
- 0078 APPLESOFT: END OF PROGRAM PTR
- 0079 APPLESOFT: END OF PROGRAM PTR
- 007A APPLESOFT: END OF PROGRAM PTR
- 007B APPLESOFT: END OF PROGRAM PTR
- 007C APPLESOFT: END OF PROGRAM PTR
- 007D APPLESOFT: END OF PROGRAM PTR
- 007E APPLESOFT: END OF PROGRAM PTR
- 007F APPLESOFT: END OF PROGRAM PTR

**EXTERNAL LABEL ADDRESSES**********

- 0100 START OF 6502 STATS
- 0200 KEYBOARD INPUT IPC
- 0300 POWERON RESET IPC

**BI CLOSE AL PAGE**********
9A2F "#" CHARACTER? (9F98) NEXT OBJECT ADDR: 9A2F
9A32 NO... >>9A54
9A34 ELSE, SAVE X REG (BE3F)
9A36 CHECK STACK FOR $DB12 AS RETURN ADDR
9A38 (APPLESOFT TRACES, PRINTING #LINENO)
9A44 NOT TRACING? >>9A56
9A46 ELSE, SET DEFERRED MODE=4
9A48 GET SET TO PRINT THE "#" (9F98)
9A4E RESTORE X REG (BE3F)
9A51 AND GO TO OTHER OUTPUT HANDLER >>B849 (0103)

9A54 NOT A #, SAME AS LAST OUTPUT TOO? (BE4)
9A57 (SAVE FOR NEXT TIME THRU) (BE4C)
9A5A NO, ALL IS WELL >>9A74
9A5C TWO RETURNS IN A ROW?
9A5E NO, ALL IS WELL >>9A74
9A68 HAS HORIZONTAL CURSOR POSN CHANGED?
9A6C YES... >>9A69
9A6D ELSE, ANYTHING IN PATHNAME BUFFER? (BC)
9A6F (MUST BE ALPHA)
9A70 RESTORE A REG
9A72 PATHNAME IS THERE... >>9A74
9A76 ELSE, WE ARE RECURSING INFINITELY, EX
9A78 WE WERE'NT TRACING AFTER ALL, RESTORE
9A79 AND A REG, THEN FALL THRU TO EXIT (FBD)

9A74 *********** ECHO OUTPUT CHAR AND EXIT ***

9A74 PUT BACK REAL CSWL/KSWL VECTORS <AAD0>
9A77 OUTPUT THE CHARACTER <FD1E> X (BE3F)
9A79 WAS IT A RETURN? 98
9A7C NO, EXIT NOW >>9ABD
9A7E ELSE, WAS APPLESOFT TRACING?
9A82 YES >>9ABB
9A84 NO, CLEAR MY TRACE FLAG (PSEUDO TRACE)
9A87 FORCE APPLESOFT TO TRACE FOR MY BENEFIT
9A88 RESTORE A REG AND FALL THRU TO EXIT BI

9ABD *********** SAVE ACTUAL IN/OUT VECTORS **

9ABD ---
9A88 COPY KSWL/H TO VECIN
9A98 AND CSWL/H TO VECOUT
9A9A IN BI GLOBAL PAGE (BE31)

***************
<table>
<thead>
<tr>
<th>ADDRESS</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>9B1D</td>
<td>AND CLOSE FILES AND QUIT SYSL</td>
</tr>
<tr>
<td>9B20-9B7F</td>
<td>ERROR HANDLER ****</td>
</tr>
<tr>
<td>9B20</td>
<td>ERROR=3, &quot;NO DEVICE CONNECT</td>
</tr>
<tr>
<td>9B22</td>
<td>MAIN ENTRY: STORE ERROR CODE $9F</td>
</tr>
<tr>
<td>9B25</td>
<td>AND IN APPELSOF ONERR</td>
</tr>
<tr>
<td>9B27</td>
<td>CHECK BI STATE (BE42)</td>
</tr>
<tr>
<td>9B2A</td>
<td>MEMORIZE WHETHER IT'S IMMED</td>
</tr>
<tr>
<td>9B2F</td>
<td>SET A HIGH FILE LEVEL FOR HD?</td>
</tr>
<tr>
<td>9B34</td>
<td>NO ACTIVE READ/WRITE FILES 3 (BE0F)</td>
</tr>
<tr>
<td>9B3D</td>
<td>CLOSE ALL OPEN FILES AT OR</td>
</tr>
<tr>
<td>9B40</td>
<td>FILE LEVEL = 30F</td>
</tr>
<tr>
<td>9B42</td>
<td>ML1: CLOSE (ALL) $BE70&gt;</td>
</tr>
<tr>
<td>9B45</td>
<td>ERROR? &gt;&gt;9B59</td>
</tr>
<tr>
<td>9B47</td>
<td>WRITE ANY DATA I HAVE BUFFERED</td>
</tr>
<tr>
<td>9B4A</td>
<td>ERROR? &gt;&gt;9B59</td>
</tr>
<tr>
<td>9B4C</td>
<td>PUT FILE LEVEL BACK TO ZERO</td>
</tr>
<tr>
<td>9B54</td>
<td>NOW FLUSH ALL OPEN FILES</td>
</tr>
<tr>
<td>9B56</td>
<td>ML1: FLUSH (ALL) $BE70&gt;</td>
</tr>
<tr>
<td>9B59</td>
<td>--</td>
</tr>
<tr>
<td>9B5A</td>
<td>ASSUME MODE WILL BE 4 (DEF)</td>
</tr>
<tr>
<td>9B5C</td>
<td>MEMORIZE WHETHER BASIC ONE</td>
</tr>
<tr>
<td>9B5E</td>
<td>DEFERRED MODE CURRENTLY? &gt; &gt;</td>
</tr>
<tr>
<td>9B60</td>
<td>NO, STILL IMMEDIATE MODE (F)</td>
</tr>
<tr>
<td>9B62</td>
<td>--</td>
</tr>
<tr>
<td>9B63</td>
<td>SET MODE AS DEFINED ABOVE &lt;B60D&gt;</td>
</tr>
<tr>
<td>9B66</td>
<td>RESTORE BI'S CSWL/KSWL INTER-ACTIVE</td>
</tr>
<tr>
<td>9B69</td>
<td>GET ERROR CODE (BE0F) 9B62</td>
</tr>
<tr>
<td>9B6D</td>
<td>BASIC ONERR ACTIVE? THEN GOSUB=0</td>
</tr>
<tr>
<td>9B70</td>
<td>NO, JUST PRINT ERROR MESSAGE</td>
</tr>
<tr>
<td>9B73</td>
<td>CLOSE EXEC FILE IF ONE IS &lt;SFD&gt;</td>
</tr>
<tr>
<td>9B77</td>
<td>DEFERRED MODE? &gt;&gt;9B55</td>
</tr>
<tr>
<td>9B79</td>
<td>IMMED. MODE, PRINT RETURN</td>
</tr>
<tr>
<td>9B7C</td>
<td>WARMSTART APPELSOF &gt;&gt;D43F</td>
</tr>
<tr>
<td>9B7F</td>
<td>RESTORE STACK FOR BASIC OPEN &lt;B555&gt;</td>
</tr>
<tr>
<td>9B84</td>
<td>PASS ERROR CODE TO BASIC</td>
</tr>
<tr>
<td>9B85</td>
<td>--</td>
</tr>
<tr>
<td>9B87</td>
<td>JUMP INTO APPELSOF ERROR HANDLER</td>
</tr>
<tr>
<td>9B8A</td>
<td>RETURN TO IMMED.</td>
</tr>
<tr>
<td>9B8A</td>
<td>CLEAR APPELSOF ERRNUM</td>
</tr>
<tr>
<td>9B8E</td>
<td>WILL LOOK FOR &quot;F&quot; FROM APPELSOF</td>
</tr>
<tr>
<td>9B93</td>
<td>SET NORMAL VIDEO IN APPELSOF</td>
</tr>
<tr>
<td>9B96</td>
<td>RESTORE TRUE CSWL/KSWL (9A0H)</td>
</tr>
<tr>
<td>9B99</td>
<td>TRY TO WRITE BUFFERED DATA</td>
</tr>
<tr>
<td>9B9C</td>
<td>RESET MODE/SET UP BI'S INTER-ACTIVE</td>
</tr>
<tr>
<td>9B9F</td>
<td>RESTORE REGISTERS &lt;9A3A&gt;</td>
</tr>
<tr>
<td>9BA2</td>
<td>GO TO PROCESS IMMED. INPUT &gt;&gt; &lt;P273&gt;</td>
</tr>
<tr>
<td>9B82</td>
<td>REQUEST &gt;&gt;9ABA</td>
</tr>
<tr>
<td>BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84</td>
<td>NEXT OBJECT ADDR: 9C18</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>ADDR</td>
<td>DESCRIPTION/CONTENTS</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>9C18</td>
<td>GO PROCESS COMMAND LINE &lt;9AD9&gt;</td>
</tr>
<tr>
<td>9C1B</td>
<td>CHECK COMMAND NUMBER (BE53)</td>
</tr>
<tr>
<td>9C1E</td>
<td>IMMEDIATE EXIT? IF NOT, GET NEXT LINE &gt;&gt;9BFC</td>
</tr>
<tr>
<td>9C20</td>
<td>RETURN</td>
</tr>
</tbody>
</table>
| *
| HANDLE EXEC PROMPT > ***************** |
| 9C21 | GET SET TO READ EXEC LINE <9DB9> |
| 9C24 | READ SINGLE CHARACTER PER CALL <9C77> |
| 9C27 | NO ERRORS, EXIT TO CALLER NOW >>9C20 |
| *
| EXEC ERROR RECOVERY ***************** |
| 9C29 | CLOSE EXEC FILE <B29F> |
| 9C2C | WAS ERROR "END OF DATA"? |
| 9C2E | NO, REAL ERROR THEN >>9C42 |
| 9C30 | ELSE, OR -- JUST STOP EXECING |
| 9C32 | GET CURSOR HORIZONTAL POSITION |
| 9C34 | IF IN MID LINE, PASS SCREEN CHAR BACK >>9C3D |
| 9C36 | ELSE, CHANGE PROMPT TO "\"
| 9C3A | AND RETURN WITH A BACKSPACE |
| 9C3C | RETURN |
| 9C3D | GET SCREEN CHARACTER UNDER CURSOR |
| 9C3F | AND EXIT THRU KSWL TO GET REAL KEYPRESS >>038 |
| 9C42 | REAL ERROR, GO TO B1'S MAIN ERROR HANDLER >>9B22 |
| 9C45 | INPUT FILE ACTIVE ***************** |
| 9C4B | THEN RESET TO IMMEDIATE MODE >>9B8A |
| 9C4E | ELSE, REMOVE CURSOR FROM SCREEN (BE3E) |
| 9C53 | CHECK KEYBOARD (C000) |
| 9C56 | NO KEYPRESS? >>9C68 |
| 9C58 | GOT A KEY, IS IT CONTROL-C? |
| 9C5A | NO, IGNORE IT >>9C60 |
| 9C5C | CLEAR STROBE AND EXIT TO CALLER (C010) |
| 9C5F | RETURN |
| 9C60 | GET PROMPT AGAIN |
| 9C62 | IS THIS A DIRECTORY FILE? (BE47) |
| 9C65 | YES >>9CC4 |
| 9C67 | NO, IS PROMPT = ""? |
| 9C69 | YES, READ A SINGLE BYTE AT A TIME >>9C71 |
| 9C6B | ELSE, READ ENTIRE LINE <9C96> |
| 9C6E | ERROR? >>9C42 |
| 9C70 | RETURN |

<table>
<thead>
<tr>
<th>BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84</th>
<th>NEXT OBJECT ADDR: 9C70</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADDR</td>
<td>DESCRIPTION/CONTENTS</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>9C71</td>
<td>READ SINGLE BYTE FROM INPUT FILE &lt;9C77&gt;</td>
</tr>
<tr>
<td>9C74</td>
<td>ERROR? &gt;&gt;9C42</td>
</tr>
<tr>
<td>9C76</td>
<td>RETURN</td>
</tr>
</tbody>
</table>
| *
| READ NEXT BYTE OF FILE ***************** |
| 9C77 | SAVE CURRENT READ/WRITE COUNT (BD9) |
| 9C7A | IN L KEYWORD VALUE (BE5F) |
| 9C7F | SET UP TO READ ONE BYTE (BE9D) |
| 9C84 | MLI: READ <B7E8> |
| 9C87 | ERROR? >>9C95 |
| 9C89 | PUT COUNT BACK TO MAXIMUM AGAIN (BE5F) |
| 9C8F | GET FIRST CHARACTER ON $200 LINE (BBF7) |
| 9C92 | AND RETURN THAT TO CALLER ($B00) |
| 9C95 | RETURN |
| *
| READ NEXT LINE OF FILE ***************** |
| 9C96 | REMOVE CURSOR FROM SCREEN (BE3E) |
| 9C98 | MLI: READ <B7E8> |
| 9C9A | ERROR? >>9C95 |
| 9C9C | GET LENGTH ACTUALLY TRANSMITTED (BED8) |
| 9C9E | NOTHING? >>9CB0 |
| 9C9F | GOT SOMETHING, FIND END OF DATA (BED7) |
| 9CA4 | FETCH LAST BYTE OF LINE (B1FF) |
| 9CB1 | IS IT A RETURN CHARACTER? |
| 9CB3 | NO, LEAVE LINE ALONE >>9CBD |
| 9CB5 | YES, WAS L KEYWORD GIVEN? (BE57)? |
| 9CBA | YES, LEAVE IT BE >>9CBD |
| 9CC0 | ELSE, CHOP OFF THE RETURN ITSELF |
| 9CBD | AND EXIT WITH A RETURN |
| 9CBE | RESTORING Y REG AS YOU GO (BE40) |
| 9CC3 | RETURN |
| *
| READING DIR FILE ***************** |
| 9CC4 | "\" PROMPT? |
| 9CC6 | YES, EXIT RIGHT NOW >>9CBD |
| 9CC8 | ELSE, REMOVE CURSOR FROM SCREEN (BE3E) |
| 9CCD | SET 80 COLUMNS |
| 9CD4 | MLT: GET MARK <B7E8> |
| 9CD7 | ERROR? >>9D4B |
| 9CD9 | ARE WE AT BEGINNING OF THIS FILE? (BC8) |
| 9CDF | NO, CONTINUE >>9D0E |
| 9CE1 | YES, CAT FLAG = 2 |
| 9CE6 | READ DIRECTORY HEADER <B1B7> |
| 9CE9 | ERROR? >>9D48 |
| 9CF6 | SET THE L VALUE OF THIS DIR FILE IN (BCCF) |
Beneath Apple ProDOS Supplement

BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84  NEXT OBJECT ADDR: 9CF9

9CF9 THE OPEN FILE LIST TO THE ENTRY LENGTH (BC88)
9CF6 AND THE NUMBER OF ENTRIES PER BLOCK (BD96)

********** FORMAT DIRECTORY NAME **********

9CF7 GO FORMAT NAME OF DIRECTORY <B112>
9D02 STORE THE LENGTH OF LINE AT $200
9D07 PUT A RETURN CHAR AT END OF LINE
9D0C AND EXIT TO CALLER
9D0D RETURN

9D0E GET CAT FLAG (BE4F)
9D11 IF ZERO, GO PROCESS INDIVIDUAL ENTRIES >>9D51
9D13 IF MINUS, GO DO SUMMARY LINE OR EXIT >>9D28
9D15 POSITIVE, ASSUME NULL LINE WANTED
9D17 DROP CAT FLAG BY ONE (BE4F)
9D1A IF ZERO, JUST GO PRINT A BLANK LINE >>9D02

********** FORMAT TITLE LINE **********

9D1C ELSE, BLANK OUT $200 AND <A6A9>
9D21 UNPACK "NAME TYPE BLOCKS ETC... <9FE7>

9D24 LINE LENGTH IS 80
9D26 GO RETURN IT TO CALLER >>9D02

********** FORMAT SUMMARY LINE **********

9D28 DO SUMMARY LINE?
9D2A NO, JUST EXIT (ALL DONE) >>9D4E
9D2C YES, DROP CAT FLAG SO EXIT NEXT TIME (BE4F)
9D31 CLEAR READ/WRITE COUNT (BED9)
9D39 MIL: READ <B70>
9D3C FORMAT BLOCKS FREE AND IN USE SUMMARY LINE <B141>
9D40 GET REF NUM (BED6)
9D43 AND COPY TO GET/SET LIST (BEC7)
9D47 NO ERRORS, EXIT >>9D24
9D49 ERROR, JUMP TO BI ERROR EXIT >>9D4E
9D4B "END OF DATA" ERROR
9D4E GO TO BI ERROR EXIT >>9B22

********** FORMAT FILE/DIR ENTRIES **********

9D51 SET DIR ENTRY NUM COUNTER TO -1
9D56 GET REF NUM (BED6)
9D59 *12
9D5B USE AS INDEX TO GET ENTRY LENGTH (BCFF)
9D64 AND Entries PER BLOCK FROM OPEN FILE LIST (BD96)
9D6A POSITION ON EVEN BLOCK BOUNDARY (BEC9)
9D70 AND GET SECTOR OFFSET (BEC8)
9D74 SKIP FILE/DIR ENTRIES UNTIL POSITIONED TO (BCBB)

9D77 CURRENT POSITION IN THIS BLC
9D7F READ NEXT DIR ENTRY FROM FILE
9D82 NO ERROR? >>9D90
9D84 ERROR, IF RANGE ERROR...
9D86 NO, TRUE ERROR >>9D4E
9D8B RANGE ERROR, READY FOR SUMMARY JAN 84 NEXT OBJECT ADDR: 9D77
9D8D RETURN A BLANK LINE THIS TIME

9D90 FORMAT FILE/DIR ENTRY INTO CK
9D93 AND RETURN IT TO CALLER >>9E

9D96 ********** PREFIX INPUT ACTIVE <B22B>

9D96 PROMPT = "I"?
9D98 NO, ALL IS WELL >>9D9D
9D9A YES, RETURN TO IMMEDIATE Mode LINE NEXT (BE4F)
9D9C REMOVE CURSOR FROM SCREEN ($3 >>9D02
9DAA PREFIX NO LONGER ACTIVE APT
9DAB COPY PATHNAME BUFFER (PREFIX) <A501>
9DAD TO $200 ($1FF)
9DB3 RETURN WITH IT TO BASIC (BC7)
9DB8 RETURN

********** SETUP TO READ LINE **********

9DB9 SET READ REF NUM FOR EXEC FILE >>988A
9DBF READ TO $200 ($2E)
9DC4 FOR SEF BYTES OF LENGTH $62 THIS (BE46)
9DC9 (OR UNTIL A RETURN CHAR) $5 (BCBC)
9DD1 RETURN

9DD2 ********** OUTPUT INTERCEPT: (LOOK FOR CONTROL-d)

9DD2 SAVE REGISTERS <9F99>
9DD5 PRINTING A CONTROL-d? FILE (B2A3)
9DD7 NO >>9DF0
9DD9 YES, WRITE OUT ANY BUFFERED
9DDC NOTHING IN COMMAND LINE (BE)
9DDF READ FILE INACTIVE (BE44)
9DE2 WRITE FILE INACTIVE (BE45)
9DE5 PREFIX READ INACTIVE (BE46) MODE = C
9DF0 SET MODE = 0 FROM NOW ON <<
9F0A RESTORE REGS AND EXIT >>9FA

9DF8 GOT A CONTROL-d...
9DF2 SET MODE = 4 FROM NOW ON <<
9DF5 RESTORE REGISTERS <9F99> DATA <A2B2>
9DF8 OUTPUT CHARACTER AND EXIT >>B
***** OUTPUT INTERCEPT: MODE = 8  ***********************

9DFB ***** (ASSEMBLE COMMAND LINE)

9DFB SAVE CHAR IN COMMAND LINE (9200)
9E01 SWP IT A RETURN?
9E04 WAS READY TO ROLL >9E16
9E06 YES BUMP CHARACTER COUNTER (BE4B)
9E08 NOW EXIT TO CALLER >9E12
9E08 ASM LINE TOO LONG
9E0D 0?UNITX ERROR" >>9E22
9E0F" 5HE RESTORES X REG AND EXIT (BE3F)
9E12 RETURN
9E15 RF

9E16 <<L LINE? >>9E25
9E18 WITH PUT BACK TRUE CSWL/KSWL <9A00>
9E1A NOUTX SCAN CMD LINE <A6B4>
9E1D ERROR? >>9E0F
9E20 ERF PUT BACK BI'S INTERCEPTS <9ABD>
9E22 NO
9E25 8=4 NOW <9FA0>
9E27 MSTORE REGS AND EXIT >>9FA3
9E2A RF

***** WRITE BUFFERED CHARACTER ***********************

9E2D ***** (Y REG (BE40)

9E2D SAVX PROMPT
9E30 CHECK TO SEE IF WE ARE IN "IF", >>9E40
9E32 CHINT", "LIST", OR "CALL" STATEMENTS >>9E40
9E34 PRINT APPLESOFT PROGRAM >>9E40
9E36 IF NOT, EXIT TO CALLER... (BE40)
9E3A IS THE CHARACTER ECHOED TO SCREEN >>9A74
9E3D YES

9E40 GROVE INTO BUFFER JUST ABOVE HIMEM
9E45 SGOF INDEX (BE4A)
9E4A HB" >>9E5A
9E4D OF ENUM, SAVE REGISTERS <9F99>
9E55 BUFFER BUFFER OUT TO DISK >9E25
9E52 WROR? >>9E0F
9E55 ERFSTORE REGISTERS <9FA3>
9E57 READ EXIT ANYWAY
9E5A A

***** OUTPUT INTERCEPT: MODE = 4  ***********************

(INITIAL ENTRY FOR A RUNNING PROGRAM)
(FLUSH OUT NON COMMAND LINES)

9E5B ***** (BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84 NEXT OBJECT ADDR: 9DF8)

9E5B PRINTING A ":"? (9F98)
9E5E NO >>9E78
9E60 YES, SAVE X REGISTER (BE3F)
9E64 RETURN ADDR IS IN APPLESOFT... (0103)
9E67 TRACE ROUTINE...
9E68 AT SS012? (0104)
9E70 YES >>9E55
9E72 NO, RESTORE REGISTERS (9F98)
9E78 IS WRITE FILE ACTIVE? (BE45)
9E7B NOPE >>9E9B
9E7D YES, PRINTING A ":"?
9E7F NO >>9E85
9E81 YES, SAME AS PROMPT CHARACTER?
9E83 YES >>9E85
9E85 NO, PRINTING A RETURN CHAR?
9E87 NO >>9E2D
9E89 YES, GET PROMPT
9E8F DOES IT INDICATE RECURSION? >>9E2D
9E91 YES, WRITE BUFFER OUT <A82B>
9E94 OUTPUT FILE INACTIVE NOW (BE45)
9E99 EXIT WITH RETURN CHAR >>9ECE
9E9B
9E9C INPUT FILE ACTIVE? (BE44)
9EA2 NO >>9EAC
9EAA YES, CHECK PROMPT
9EAE OR IN 084
9EAF CONTROL-D?
9EAA YES >>9ED1
9EAC ---
9EAD NO, HOW BOUT "."?
9EAF NO, EXIT WITH ECHO THEN >>9ECE
9EB1 YES, DO THIS THE PROMPT CHAR?
9EB3 NO, EXIT WITH ECHO >>9ECE
9EB5 YES, SAVE REGISTERS <9F99>
9EB8 CHECK OPEN FILE COUNT (BE4D)
9EBB NONE OPEN? >>9ECB
9EBD SOME OPEN, WRITE BUFFER OUT <A82B>
9EC0 INDICATE WRITE FILE INACTIVE NOW (BE45)
9ECD SET TRUE CSWL/KSWL <9A00>
9ECE PRINT "FILE(S) STILL OPEN" <BE0C>
9ECB RESTORE REGS <9FA3>
9ECE AND ECHO EXIT >>9A74

9ED1 ---
9ED2 CHAIR IS A RETURN?
9ED4 NO >>9ED9
9ED6 YES, SAME AS LAST CHAR OUTPUT? (BE4C)
9ED9 (SAVE IT FOR THIS TEST NEXT TIME) (BE4C)
9EDC NOT SAME, NO PROBLEM THEN >>9E98
****** SAVE CALLERS REGISTERS ******

SAVE A,X AND Y REGS (BE3E)

RETURN

****** RESTORE CALLERS REGISTERS ******

RESTORE A,X AND Y REGS (BE3E)

RETURN

----------

BASIC Interpreter

ADDRESS: 9EE1

DESCRIPTION: SCUZZER SOFT TRACING INTERCEPT

**PROMPT FOR EXECUTION**

BUMP A Registers: Here for every statement

9E85 --- PRINT INHIBITED (WHILE PROG is ACTIVE)

MARK A

9E8B JUST COMPLETED LINE POINTER

9E8D RESTORE CALLER

9E8F DOES AGREEND EVENTS IN A Recursion

9EE3 YES, 32A0 is SOFTSTORE's stack

9E85 ELSE, 32A0 is HERE (BE41)

9E89 IS IT PICK UP TRACED DATA >>9F68

9E90 OR ELSE 32A0 is HERE (BE41)

9E92 COMPLETED AT THIS POINT >>9F1D

9E96 AT LEAST 32A0 is SPACE CTR (BE49)

9E98 YES AT 32A0 is SPACE CTR

9E9A NO, WE NEED A N2 FREESPACE IN PAGES

9E9C DIALOGUES AVAILABLE

9EA0 COMPLETED

9E94 ESSAYED DATA (A62B)

9EA5 GET NEXT IMAGE COLLECTOR (A67B)

9EB1 --- EXTEND SPACE NOW

9EB0 JUMP TO STRING SPACE CTR (BE49)

9ED8 STORE BACK TEXT

9EFC LOOK AT TOP OF APPLESOFT TO EXECUTE IT >>D820

9EF0 IT IS 32A0 ON PROMPT

9EB2 AND JUMP TO THIS TOKEN TABLE (BE73)

9EF2 IF ON CLEAR PRINT ONE OF THE FOLLOWING: >>9F2D

9F00 GO TO 32A0 ON PROMPT = 0

9F03 BEGIN OF 32A0 ON PROMPT CARRY 0

9F0B NOW ON TAFIST CHAR 32A0 is AREA (BE4C)

9F08 GO NEXT TIME THRU (BE5D)

9F10 LIST: STRING FOR COMMANDS (BE3E)

9F12 (DON'T PRINT THE IF CR PRINT >>9F50

9F14 GO DO IT >>9F5D

9F16 CALL FOR COMMANDS NOW

9F18 (DON'T PRINT >>9F5D

9F1A GO DO IT >>9F5D

---------

SCRUZZER/CONTENTS

ADDRESS: 9F65

DESCRIPTION: SCUZZER/CONTENTS

NEXT OBJECT ADDR: 9F4A

---------

BASIC Interpreter

ADDRESS: 9F4C

DESCRIPTION: SCUZZER/CONTENTS

NEXT OBJECT ADDR: 9F4A

---------

BASIC Interpreter

ADDRESS: 9F50

DESCRIPTION: SCUZZER/CONTENTS

NEXT OBJECT ADDR: 9F4A
Beneath Apple ProDOS Supplement

BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84 NEXT OBJECT ADDR: 9FAC

---

ADDR DESCRIPTION/CONTENTS
---

9FAD ********** SET MODE AND CSWL/KSWL **********

9FAD' STORE "STATE" MODE FROM X REGISTER (BE42)

9FB2 COPY PROPER CSWL/KSWL VALUES TO REDIRECT... (BE51)

9FB5 VECTOR DEPENDING ON CURRENT MODE (BE38)

9FBE RETURN

9FBB ********** PRINTERR: PRINT ERROR MSG **********

9FBE ---

9FC0 GET INDEX INTO PACKED MESSAGE TEXTS (BA65)

9FC3 UNPACK MESSAGE INTO $201 <9FE7>

9FC9 SAVE THE LENGTH (BC86)

9FCC SKIP A LINE <9FE2>

9FD1 PRINT A BELL <9FE4>

9FD4 ---

9FDC PRINT CONTENTS OF $201 MSG BUFFER ($201)

9FE2 PRINT A RETURN CHARACTER

9FE4 AND EXIT >>FDED

9FE7 ********** UNPACK ERROR MESSAGE **********

9FE7 NOTHING IN BUFFER AT FIRST

9FED GET A NIBBLE FROM PACKED MSG <A009>

9FF8 NON-ZERO, COMMON CHARACTER >>9FF7

9FF2 IF ZERO, GET NEXT NIBBLE <A009>

9FF5 AND CONVERT TO UNCOMMON CHAR INDEX

9FF7 ---

9FF8 GET THE LETTER THIS NIBBLE REPRESENTS (BA7A)

9FFB ZERO? THEN END OF MESSAGE >>A008

9FFD GET INDEX INTO OUTPUT BUFFER (BE4B)

A000 AND STORE THE CHARACTER THERE ($201)

A003 BUMP INDEX (BE4B)

A006 AND CONTINUE >>9FED

A008 RETURN

A009 ********** UNPACK MESSAGE BYTE **********

A009 GET NEXT MSG BYTE (BA9A)

A00C WORKING ON SECOND NIBBLE? >>A020

A00E NO, TAB INDICATOR? >>A016

A010 NO, ISOLATE HIGH NIBBLE

A014 NEXT TIME GET LOW NIBBLE

A015 RETURN

A016 ---

A017 GET TAB POSITION (BA9A)

A01A AND BUMP OUTPUT PTR ACCORDINGLY (BE4B)

A01E THEN GO BACK FOR NEXT NIBBLE >>A009

---

BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84 NEXT OBJECT ADDR: A01E

---

ADDR DESCRIPTION/CONTENTS
---

A020 BUMP BYTE PTR FOR NEXT TIME

A021 ISOLATE LOW NIBBLE

A023 NEXT TIME GET HIGH NIBBLE

A024 RETURN

A025 ********** WRITE ONE BUFFERED BYTE **********

A025 SET UP COUNT OF A001

A029 AND JUMP INTO ROUTINE BELOW >>A03E

A02B ********** WRITE BUFFERED DATA/TEST ERROR **********

A02B WRITE BUFFERED DATA <A037>

A02E OK? THEN EXIT >>A053

A031 ERROR?, POP OUT OF THIS SUBROUTINE

A034 AND GO TO ERROR HANDLER >>B22

A037 ********** WRITE ALL BUFFERED DATA **********

A037 ---

A039 GET BUFFERED DATA COUNT (BE4A)

A03C NONE BUFFERED? >>A052

A03E STORE BUFFERED DATA COUNT IN RW FARMS (BE9D)

A046 MLI: WRITE <BE70>

A04C NOTHING BUFFERED NOW, COUNT=0 (BE4A)

A050 ERROR? >>A053

A052 NO, EXIT

A053 RETURN

A054 ********** SPECIAL GARBAGE COLLECT **********

(PULL OUT STRING CONSTANTS ALSO)

A054 DO GARBAGE COLLECTION NORMALLY FIRST <A07B>

A057 ERROR? >>A07A

A05B START OF STRING AREA = PROGRAM START PTR (BC84)

A063 USE GENERAL PURPOSE BUFFER (ABOVE HIMEM)

A065 FOR A GARBAGE COLLECT WORKAREA (BC7D)

A06A IT IS 3+1 PAGES IN LENGTH (BC7E)

A06F END OF STRING AREA IS AT END OF FREE AREA (BC86)

A077 GO COLLECT CONSTANT STRINGS NOW <<A0C2

A07A THEN EXIT

A07B ********** "FRE" COMMAND **********

(FAST APPELSOFT STRING GARBAGE COLLECTION)

| GENERAL PURPOSE BUFPR |
| (TOP OF OLD STRINGS) |
| HIMEM --> |
| NEW STRINGS BUILDING |<--- |
A07B STRING AREA START IS ON PAGE BOUNDARY
A082 ASSUME 4 PAGE WORKAREA (BC7E)
A087 IN GENERAL PURPOSE BUFFER ABOVE HIMEM (BC7D)
A08C STRING START PTR IS START OF STRING AREA (BC84)
A090 COMPUTE NUMBER OF FREE PAGES
A092 AT LEAST ??
A094 IF NOT, USE G.P. WORKAREA INSTEAD >>A0B0
A096 DON'T USE ALL OF FREE AREA (LEAVE $300)
A098 NEW WORKAREA SIZE IS FREE AREA SIZE-$300 (BC7E)
A09D SET PTR TO WORKAREA AT FIRST FREE PAGE
A0A4 COMPUTE NUMBER OF STRING PAGES
A0A8 USE SMALLER OF STRING PAGES OR WORKAREA SIZE (BC7E)
A0AD AS NEW WORKAREA SIZE (BC7E)
A0B0 END OF STRING AREA IS HIMEM
A0B8 JUMP TO NEXT INSTRUCTION >>A0BD
A0BD STRING START LSB IS HIMEM INITIALLY (BC85)
A0C2 RECORD WHETHER LAST PAGE IS PARTIAL
A0C6 STRING START MSB IS HIMEM INITIALLY (BC86)
A0CB ADJUST LONGAGE AND HIRANGE MSB'S
A0CD FOR PARTIAL PAGES AT EITHER END, (BC7F)
A0D0 SETTING THEM AT HIMEM FOR NOW.
A0D9 SET UP ARRAY END MSB +1 FOR COMPARES (BC82)
A0DC $3B-$3F >> FIRST VARIABLE (LESS 7 BYTES)
A0DB (EACH VARIABLE IS 7 BYTES)
A0E8 SET UP ARRAY START LSB FOR COMPARES
A0ED GET LORANGE VALUE (BC7F)
A0F0 PRIOR TO STRING AREA? (BC84)
A0F3 YES, THEN DONE! >>A133
A0F5 ELSE, DROP LORANGE BY WORKAREA SIZE (BC7E)
A0F8 AND SAVE THIS VALUE (BC7C)
A0FD NOW DROP IT ALSO BY THE DISTANCE BETWEEN
A10D THE OLD LORANGE AND THE STRING START PTR (BC7F)
A107 USE THE LOWER OF THE TWO VALUES (BC7C)

A10C PRODUCE THE MAXIMUM SIZED RANGE (BC7C)
A112 IS IT BELOW THE BOTTOM OF THE STRINGS? (BC84)
A114 >>A119
A117 USE THE BOTTOM POINTER INSTEAD (BC84)
A119 COUNT FOR PARTIAL PAGE
A11C NEXT FINAL LORANGE VALUE (BC7F)
A11F SAME PAGES BELOW HIRANGE TO WORKAREA <A1D2>
A121 LOAD ROOM FOR NEW STRINGS
A124 SET SIMPLE STRING VARS FOR THIS RANGE <A134>
A126 >>A131
A129 COLLECT STRING ARRAYS <A16A>
A12F RANGE = OLD LORANGE (BC7F)
A131 SUB LOOPTING >>A0D0
A133 "ERROR, "RAM TOO LARGE"
A134 *** COLLECT STRING ARRAYS
A135 T
A137 >>A16A
A13F ARE ARRAYS NOW?
A145 NA OR >>A168
A147 WE ARE DONE >>A168
A14E NA OR A134
A150 NA OR A134
A154 NA OR A134
A158 TEST SUB STRING POINTER
A15B >>A134
A160 >>A134
A162 >>A167
A165 ANYTHING NOT OUT AND TACK IT TO HIMEM <A1F5>
A167 NEXT WELL, GET NEXT VARIABLE >>A135
A168 ERROR, EXIT NOW
A169 EXIT CALLER

A16A *** COLLECT SIMPLE STRINGS
A16D THE NEXT ARRAY <A199>
A16F >>A199
A173 GET MSB OF ITS STRING PTR
A176 >>A199
A179 WITHIN MY RANGE? (BC7F)
A17B >>A193
A17D >>A193
A180 ANYTHING NOT OUT AND TACK IT TO HIMEM <A1F5>
A182 CONTINUE WITH NEXT ARRAY ELEMENT >>A184
A183 ERROR EXIT
A183  ---
A184 BUMP POINTER TO NEXT ARRAY MEMBER
A185* POINTER NOW AT NEXT ARRAY? (BC81)
A191 NO, DO THIS ELEMENT >>A16F
A197 YES, SET UP TO PROCESS THAT ONE THEN >>A16A
A199 ********** FIND NEXT STRING ARRAY **********************
A199  ---
A19A $3E -- ARRAY VARIABLES (BC81)
A1A1 AT END OF ARRAY VARS
A1A3 NO, CONTINUE >>A1A9
A1A7 YES, OUT (CARRY SET, NO MORE ARRAYS) >>A1D1
A1A9 POINT TO ARRAY FOLLOWING THIS (LSB AND...)
A1B3 MSB TO X REGISTER
A1BA CHECK TYPE OF VARIABLE
A1BF SKIP INTEGER AND REAL ARRAYS >>A199
A1C3 GET NUMBER OF DIMENSIONS
A1C5 *2 TO SKIP SIZES
A1C6 *5 TO SKIP FIXED STUFF AT BEGINNING
A1CA POINT TO FIRST ARRAY MEMBER
A1CE READY TO ROLL, $3E POINTS TO IT
A1D1 RETURN

A1D2 ********** COPY PAGES TO WORKAREA **********************
A1D2  ---
A1D7 $3A/$3B -- FIRST PAGE TO SAVE (BC7C)
A1D5 $3C/$3D -- WORKAREA (BC7D)
A1E2 COPY N+1 PAGES (SIZE OF WORKAREA) (BC7E)
A1E6  ---
A1F4 EXIT WHEN FINISHED

A1F5 ********** PULL STRING OUT ***********************
A1F5  ---
A1F8 YES, ITS STILL THERE THEN >>A201
A1FA ELSE, POINT TO SAVED STRING IN WORKAREA (BC7C)
A201 $3A/$3B -- STRING
A20C DROP STRING START PTR BY LEN OF THIS STRING
A211 UPDATE STRING'S LSB IN VARIABLE PTR
A215 FIX UP MSB OF STRING START PTR ALSO
A21A AND OF VARIABLE PTR
A21E IS THIS A NULL LENGTH STRING?

Beneath Apple ProDOS Supplement

BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84 NEXT OBJECT ADDR: A2A
addr description/contents

A230 YES, NO MOVE TO DC
A232 ELSE, COPY STRING...
A234 OUT OF FREESPACE?
A236 RETURN TO CALLER
A238 ALLOCATE
A240 CHECK FOR SPACE IN FREEPOOL
A24A IF NOT, "RAM TOO FULL"
A24E NOT ENOUGH, $3A-- PURPOSE ALLOCATE
A250 COMPUTE LENGTH OF STRING
A255 COPY OF COPY STRING DOWN
A25F SUBTRACT "N" FROM TOP FREE SPACE <A07B>
A260 ADD ALL POINTERS
A265 NEW HIMEM BECOMES S THERE?
A26A OLD HIMEM IS "N" <BB6>
A26D FIND PAGE JUST LARGE MSG
A26F RETURN
A273 ADD ALL POINTERS
A277 "FREE PAGE" "FREE SPACE"
A27B "FREE STRING" "FREE COPY" STRINGS FOR COPY
A289 ********** FREE BUFF "N" PAGES IN MEMORY <A396>
A28C PREPARE ADDRESS MSG'S (BB6)
A28E GARbage COLLECT in SIMPLE & ARRAY VARS <A3DA>
A292 IF HIMEM $100 PAGES LOWER
A296 AND HIMEM$400 IN MOST A FILE BUFFER (BC88)
A29C (COPY all's)
A2A2 $32 -- LENGTH OF S
A2A6 COPY STRINGS UP 4
A2BF FREE TO FREE STRING OF STRING <A87B>
A2C2 FREE, DONE? >>A2BE
A2C5 CHECK OPEN FILE $O $3A/$3B
A2C9 OPEN FILE? (HOW CA
A2CA WHICH FILE'S BUFFER
A2CF SEARCH UNTIL IT'S PAGES <A3BA>
A2DC ADDS UP BY $400 <A3DA>
A2DE HUNT (BE40) IN THAT BE7?) >>A2D4
A2E9 IS NEXT TO HIMEM? FOUND... >>A2D5
BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84  NEXT OBJECT ADDR: A2D4

ADDR DESCRIPTION/CONTENTS

A2D4 RETURN IF NO FILE IS USING THIS BUFFER
A2D5 ---
A2D6 GIVE THAT FILE THE BUFFER PASSED TO US (BEC9)
A2D9 (SURE HOPE THAT FILE WAS FLUSHED!) (BC93)
A2E4 PASS FILE REF NUM TO MLI (BEC7)
A2EB MLI: SET NEW BUFFER <BE79>
A2EC ERROR? >>A2D4
A2EE ---
A2EF RETURN

A2F0 Getbufr: GET A BUFFER

This routine is called through an external entry point in the global page. It allocates a fixed location buffer between the BI and its buffers.

A2F0 ALLOCATE A BUFFER OF ANY SIZE (A= PAGES) <A234>
A2F3 ERROR? >>A33A
A2F9 FIND FIRST PAGE OF BUFFER (BB9)
A2FF GET FILE OPEN COUNT (BE4D)
A302 NONE OPEN? >>A325
A304 BUMP BUFFER PTR BY $400 (BB8)
A306 TO POINT TO PREVIOUSLY ALLOCATED A30A BUFFER (BB8)
A30D FIND OPEN FILE WITH THIS BUFFER (BC93)
A312 GOT IT, (BEC9)
A315 SET FILE BUFFER REAL LOW IN MEMORY <A38D>
A318 THEN SET IT TO NEW BUFFER LOCATION <A2D6>
A31B BELOW ALL OTHERS (BEC9)
A322 DO THIS FOR EACH OPEN FILE...
A323 THEREBY INSERTING A BLANK BUFFER >>A30D
A328 IS EXEC FILE ACTIVE? (BE43)
A32B NO, DONE >>A33A
A32D YES,
A32F MOVE EXEC BUFFER DOWN ALSO <A38D>
A338 AND BUMP UP ABOVE IT A33A EXIT TO CALLER
A33B RETURN

A33C Freebufr: FREE BUFFER

This routine is called through an external entry point in the global page. It frees a fixed location buffer previously allocated by Getbufr.

A33C GET COUNT OF OPEN FILES (BE4D)
A340 INDEX THIS BY 4 PAGES PER FILE
A341 ADD TO HIMEM MSB
A344 SAVE THIS AS TOP OF BUFFERS (BB8)
A348 THEN SET UP BOTTOM AS HIMEM MSB (BB9)
A34B GET OLD ORIGINAL HIMEM (BEFORE ANY BUFFERS) (BEFB)

A34E SAME AS THIS ONE?
A350 THEN NOTHING ELSE TO DO >>A388
A352 ASSUME NO BUFFERS BY REPLACING OLD HIMEM
A354 ANY EXEC FILE OPEN? (BE43)
A357 NO, CONTINUE >>A35E
A359 YES, MOVE EXEC BUFFER TO OLD HIMEM <A32D>
A35C AND GO MOVE HIMEM DOWN BY $400 >>A37C
A35E ELSE, START WITH TOP BUFFER (BB8)
A361 ANY OPEN FILES? (BE4D)
A364 IF NOT, WE ARE DONE >>A388
A366 SEARCH FOR OPEN FILE WITH THIS BUFFER (BC93)
A369 NOT IT? >>A385
A36B GET IT, GIVE IT NEW HOME AT HIMEM
A36D AND SET BUFFER LOW <A38D>
A370 THEN TO NEW LOC <A2D6>
A374 DROP TOP BUFFER PTR BY $400 (BB8)
A37C AND DROP HIMEM BY $400
A383 AND GO DO NEXT BUFFER >>A35E
A385 ---
A386 (LOOP TO SEARCH FOR OPEN FILES) >>A366
A388 WHEN FINISHED, GARBAGE COLLECT <A8BD>
A38C ---
A38C THEN EXIT NORMALLY TO CALLER

**** SET BUFFER BELOW ALL OTHERS ***
A38D ---
A38E USE BOTTOM BUFFER PTR (BB9)
A391 SET FILE BUFFER <A2D6>
A395 AND EXIT

A396 Copy Block Down In Memory

A396 COPY ALL FULL PAGES DOWN TO THEIR NEW HOME
A39D COPYING $3A->$3C
A39A BUMP BOTH MSB'S
A39A DROP PAGE COUNTER (BC93)
A39B AND CONTINUE >>A39D
A39C NO SHORT LAST PAGE? (BC92)
A39B THEN EXIT NOW >>A389
A392 ELSE, COPY PARTIAL PAGE
A39B THEN EXIT

A38A Copy Block Up In Memory

A38A PARTIAL PAGE? (BC92)
A38B NO, JUST COPY FULL PAGES NOW >>A3C6
A38F YES, COPY SHORT PAGE FIRST <A3D1>
A3C2 DROP BOTH MSB'S
A3C2 PAGE COUNT GONE TO ZERO? (BC93)
A3C7 YES, DONE >>A3D9
BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84  NEXT OBJECT ADDR: A3CB

A3CB ELSE, DROP PAGE COUNT (BC93)
A3CE AND GO COPY A FULL PAGE UP >> A3BF
A3D1    ---
A3D2 COPY REMAINDER OF PAGE UP (BACKWARDS)
A3D9 RETURN

A3DA ********** ADJUST ALL STRING ADDR ***********************
(BC87 HAS ADDITIVE ADJUSTMENT FACTOR)

A3DA USE LOMEM PAGE AS MSB FOR $3E/$3F
A3DB GET LOMEM LSB
A3E8 AND END OF SIMPLE VARS PAGE
A3E3 JUMP INTO THE LOOP >> A3EA
A3E5 ---
A3E6 SKIP ONE SIMPLE VARIABLE
A3EA ---
A3EC OVERFLOW? >> A3F0
A3E8 YES, BUMP MSB
A3F0 FINISHED WITH SIMPLE VARS?
A3F4 (CHECK BOTH MSB AND LSB OF PTR)
A3F6 ---
A3F7 YES... >> A40D
A3F9 NO,
A3FB LOOK AT A SIMPLE VARIABLE
A400 SKIP INTEGER AND REAL VARS >> A3E5
A402 (DOUBLE CHECK MSB)
A406 ITS A STRING, POINT TO ITS LEN/ADDR
A407 ADJUST IT IF NECESSARY << A435>
A40A THEN SKIP OVER IT >> A3E5
A40D COPY ARRAYS STARTING LSB
A40F (MSB IS IN X REGISTER NOW) (BC81)
A411 ---
A413 FIND A STRING ARRAY << A199>
A416 NO MORE? THEN DONE... >> A434
A418 ---
A41B ADJUST ITS ADDRESS IF NEED BE << A435>
A421 SKIP TO NEXT STRING ELEMENT OF ARRAY
A429 AT END OF THIS ARRAY YET? (BC81)
A42C NO... >> A418
A42E (CHECK MSB ALSO)
A432 YES... GO GET NEXT ARRAY >> A412
A434 RETURN

A435 ********** ADJUST A STRING ADDRESS ***********************

A435 GET STRING LENGTH
A437 IGNORE NULL STRINGS >> A448
A439 POINT TO MSB OF ADDRESS
A43B IS STRING STORED OUTSIDE OF PROGRAM?
A43F? NO, LEAVE IT ALONE >> A448
A441 STORE ABOVE LOMEM, ADD FACTOR TO MSB
A448 THEN EXIT

A449 ********** COMPRESS ALL ADOPT 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>

A449 GARBAGE COLLECT FIRST << A854>
A44C ERROR? >> A4AE
A44E COMPUTE LENGTH OF SIMPLE AND ARRAY VARS
A453 AND SAVE IT (BC89)
A463 NEXT, COMPUTE LENGTH OF SIMPLE VARS ONLY
A467 AND SAVE IT (BC8B)
A471 SUBTRACT VAR LENGTH FROM STRING START
A473 TO FIND A PLACE TO PUT THE VARS UNDER (BC92)
A476 THE STRINGS (START ON AN EVEN PAGE BOUND)
A47C $3C/$3D --> PLACE TO PUT VARS
A483 $3A/$3B --> START OF VARS (ROUNDED TO EVEN
A485 PAGE ALIGNMENT)
A48B COPY VARS UP AGAINST STRINGS << A3BA>
A490 STORE START OF VARS PTR (BC8E)
A49B BUMPING PAGE NUMBER BY ONE
A4A0 SUBTRACT THIS PTR FROM HIMEM TO COMPUTE (BC90)
A4A3 TOTAL LENGTH OF COMBINED VARS/STRINGS
A4A5 AND SAVE THIS TOO (BC8D)
A4A8 ALSO, SAVE HIMEM MSB IN CASE THEY ARE MOVED
A4AE DONE, EXIT
BASIC Interpreter (B1) -- V1.0.1 -- 1 JAN 84  NEXT OBJECT ADDR: A4AE

**ADDRESS/DESCRIPTION/CONTENTS**

A524 LOOK UP FILE TYPE IN TABLE (B9DB)
A525 FOUND IT? >>A534
A526 FILE TYPE NOT IN MY TABLE
A527 PRINT IT IN HEXADECIMAL (A64F)
A528 AND CONTINUE BELOW >>A57A
A534 ELSE, FOR KNOWN TYPES
A535 COPY NAME OF TYPE TO THE LINE (B9E9)
A542 $80 COLUMNS PER LINE? (BCB6)
A545 YES... >>A598
A547 NO,
A549 BIN FILE?
A54B YES... >>A562
A54D TXT FILE?
A54F NO... >>A578
A551 YES, R VALUE GIVEN AS SUBTYPE
A55C CONVERT R VALUE TO DECIMAL (A66C)
A55F SKIP OVER BIN CODE >>A571
A562 BIN FILE, USE AD VALUE AS SUBTYPE
A56A CONVERT IT TO TWO HEX DIGITS (A64F)
A573 ADD AN "=" SIGN
A578 COPY MB OF END OF FILE MARK (B270)
A586 CONVERT LOW TWO BYTES OF EOF (A66C)
A58D DO CREATION DATE/TIME <<ADDE>
A590 --
A59B CONVERT BLOCKS USED <<ADDE>
A59D CHECK FOR WRITE ACCESS
A5AD UNLOCKED? >>A5A9
A5A4 NO, ADD A "="
A5A9 FALL THRU TO DO LAST MODIFIED DATE/TIME
A5AB AND THEN EXIT TO CALLER

A5AD ********** FORMAT A DATE/TIME **************

- X = OFFSET FROM $259 TO FIELD
- Y = $201 OFFSET TO DATE/TIME VALUE

A5AD ISOLATE YEAR ($25A)
A5B1 AND STORE IT (BCB5)
A5B8 ISOLATE DAY
A5BA AND STORE IT (BCB4)
A5BE ISOLATE MONTH
A5C4 (MONTH = 0 IS NO GOOD) >>A5E8
A5C8 (MONTH > 12 IS ALSO BAD) >>A5E8
A5CA STORE MONTH (BCB3)
A5CE MULTIPLY MONTH INDEX BY 3 (BCB3)
A5D1 AND SAVE IT INSTEAD (BCB3)
A5E7 (DAY = 0 IS NO GOOD) >>A5F2
A5DE (YEAR MUST BE < 99) >>A5F2
A647 ************ CONVERT 2 DIGIT NUMBER **********************
(FORCE LEFT ZERO FIELD)
A647 ---
A648 ADD 100 TO FORCE SIGNIFICANCE IN TENS
A649A CONVERT IT <A66C>
A649D IGNORE 100'S PLACE
A649E RETURN

A64F ************ CONVERT TO HEX **********************
A64F ---
A650 ISOLATE LOW NIBBLE
A652 AND GO CONVERT IT FIRST <A65A>
A656 NOW ISOLATE HIGH NIBBLE
A659 AND FALL THRU TO CONVERT IT ALSO

A66A 24 BIT SHIFT (3 BYTES)
A670 CLEAR SUM (BCB)
A691 GO ROL ACCUMULATOR LEFT ONE BIT ?
A693 ALSO ROL 4TH BYTE OF ACCUM (BCB2)
A69B IF MSB > 10... (BCB2)
A6A2 ADD ONE TO ACCUMULATIVE SUM
A6A5 ---
A6A6 SHIFT 24 TIMES >>A691
A6A8 RETURN
A6A9 ---
A6B3 RETURN

A6B4 ************ SYNTAX: PARSE COMMAND LI
(ALSO EXTERNAL ENTRY FOR COMMAND)
A6B4 INIT COMMAND NUMBER TO -1
A6BB A BLANK ENDS EACH STRING (BCA9)
A6C0 AT MOST 8 CHARACTERS IN A COMMAND
A6C3 PARSE COMMAND ITSELF <A5B>
A6C6 GET FIRST LETTER (BCBD)
A6C9 MUST BE ALPHABETIC
A6CB IT IS... >>A6D4
A6CD IT'S NOT, IS IT A "-"?
A6CF YES, OK THEN... >>A6D4
A6D1 ELSE, ITS BAD - SYNTAX ERROR >>A8
A6D4 SCAN FOR COMMAND IN TABLES <AB21> RLD DIGIT (BCB0)
A6D7 BAD COMMAND? >>A6D1

g *********

Beneath Apple ProDOS Supplement

BASIC Interpreter (B1) -- V1.0.1 -- 1 JAN 84 NEXT OBJECT ADDR: A5DE
ADDR DESCRIPTION/CONTENTS

A580 OTHERWISE, BAD DATE!
A582 BACK UP 6 CHARACTERS ON LINE
A587 AND PRINT "NO DATE" (BA37)
A5F1 THEN EXIT RIGHT AWAY
A5F2 DATE OK, GET HOUR (>25C)
A5F6 AND MINUTES (>25B)
A5FB MINUTES > 60?
A5FC NO... >>A600
A5FF YES, USE ZERO MINUTES
A600 CONVERT MINUTES (LEFT ZERO FILL) <A647>
A605 THEN PRINT A "-" (>201)
A609 GET HOUR AGAIN >
A60C GREATER THAN 24 HOURS?
A60E NOPE >>A611
A610 YES, USE ZERO
A611 10 OR MORE HOURS (TWO DIGITS?)
A614 IN ANY CASE, CONVERT HOURS <A66C>
A618 IF TWO DIGITS... >>A61B
A61A IF ONE, ADJUST LINE PTR
A61B ---
A61F CONVERT YEAR (LEFT ZERO FILL) <A647>
A623 GET MONTH INDEX (*3) (BCB3)

A65A CONVERT NIBBLE TO NUMERIC ASCII
A65C >9?
A65E NO >>A662
A660 YES, CONVERT SBA-SBF TO SC1-SC6
A662 AND STORE THE RESULT (>201)
A665 BUMP LINE INDEX BACK
A666 PRECEDED WITH A $ SIGN
A66B RETURN

A66C ************ CONVERT TO DECIMAL *****
A66D A,X = NUMBER Y=INDEX TO LAST F
A66F STORE NUMBER IN ACCUMULATOR (BCB)
A672 DIVIDE BY 10 <A68A>
A674 GET DIGIT AND CONVERT IT (BCB2)
A677 a STORE IN LINE (>281)
A67D DROP LINE INDEX BY ONE
A67E IS QUOTIENT NOW ZERO? (BCB6)
A687 NO, CONTINUE UNTIL IT IS >>A672
A689 ELSE, EXIT

******** DIVIDE ACCUMULATOR BY 1
BASIC Interpreter (B1) -- V1.0.1 -- 1 JAN 84 NEXT OBJECT ADDR: A6D9

A6D9 NO, IMMEDIATE COMMAND MODE? (BE42)
A6DC NO, DEFERRED... >>A69
A6DE IMMEDIATE, EXEC ACTIVE? (BE43)
A6F1 YES, NEVER MIND >>A69
A6F4 NO SECONDARY PATH NAME EITHER (?280)
A6FA CURRENT SLOT = DEFAULT SLOT (BE61)
A700 CURRENT DRIVE = DEFAULT DRIVE (BE62)
A705 BUFFER ALLOCATION = HIMEM (BC88)
A708 GET LENGTH OF COMMAND NAME (BE52)
A713 ...must be mon or nomon >>A776
A715 YES, IN# OR PR#?
A718 ELSE, REPARSE THE COMMAND <<A50>
A720 FOR THIS COMMAND... (BE54)
A728 DOES THE PREFIX NEED FETCHING? >>A727
A72A YES, ML: GET PREFIX FROM DEFAULT DRIVE <BE70>
A72D ---
A72F YES, NO FILENAME, LOOK FOR KEYWORDS >>A7C7
A732 "^/"
A734 YES >>A73A
A736 NO, ALPHABETIC?
A738 NO...FILE NAMES MUST BEGIN THAT WAY >>A76F
A73A ---
A73B DON'T FLUSH ANY BLANKS OUT OF PATHNAME
A748 ALLOW 64 CHARACTERS NEXT PARSE
A74C PARSE NEXT OPERAND ON LINE <AA5F>
A74A SAVE ITS LENGTH (BC8C)
A74F FOUND A PATHNAME1 (BE56)
A755 COPY PART KEYWORD TO $280 (BC8C)
A758 (ASSUMING PATHNAME1=PATHNAME2) ($280)
A75F CHECK NEXT CHAR (OTHER THAN A BLANK) <AA7A>
A762 NOT COMMA OR RETURN, BAD! >>A76C
A76A RETURN? >>A7DB
A766 NO, PATHNAME EXPECTED NOW? (BE54)
A76A YES, ALL IS WELL >>A7A2
A76C NO, "SYNTAX ERROR" >>A79
A76F NON ALPHA FILE NAME, CHECK COMMAND NUMBER (BE53)
A772 IS IT "RUN"?
A774 NO, ERROR >>A76C
A776 YES, ITS OK THEN (MIGHT BE "RUN 100") >>A7DB
A779 IN#//PR#?, REPARSE COMMAND <AA5B>
A77C RETURN (ADDRESS KEYWORD)
A77E "^/" GO PARSE THAT KEYWORD ONLY >>A7CC
A780 IF SO, $280 ACCUMULATOR <A777>
A782 ELSE, TYPING ONE BYTE'S WORTH (BCAD)
A785 CONVERT IN PR#/IN# SLOT VALUE AREA (BCEA)
A78A PUT IT IN SLOT # FOR PR#/IN# (BE56)
A78F FOUND ? SLOT # <A9A0>
A791 CONVER... >>A7A1
A795 ERROR INVERTED VALUE (BE6B)
A797 GET CO.
A79A >?? S OK >>A7D1
A79C NO, INVALID ERROR
A79E YES,?
A7A1 RETURN PATHNAME EXPECTED?
A7A2 SECOND??
A7A3 NO >>A79 Flush TO NON-BLANK <AA7A>
A7A5 YES, IF ELSE ON LINE?? >>A76C
A7A8 NOTHING, FLUSH ANY BLANKS OUT OF PATHNAME
A7AB DON'T CLOSE PATHNAME TO $281 <AA40>
A7B2 COPY $281 LENGTH (LESS 1) (0280)
A7B7 SAVE PATHNAME1 AND PATHNAME2 (BE56)
A7BC FOUND 1ST CHARACTER AGAIN <AA7A>
A7C0 GET LAST COMMA OR RETURN, "SYNTAX ERROR" >>A76C
A7C3 IF NOT >>A76D
A7C5 RETURNPATH.Flush TO NON-BLANK <AA7A>
A7CB NO, CC ERROR IF TWO COMMAS IN A ROW >>A76C
A7CA SYNTAX.KEYWORD CHAR AND PARSE ITS VALUE <A928>
A7CC LOOKUP?? >>A7A1
A7CF EXIT NO...TO NON-BLANK <AA7A>
A7D1 NO...IF ERROR IF COMMA OR RETURN NOT FOUND >>A76C
A7D4 SYNTAX.YES, GO GET NEXT KEYWORD >>A7C7
A7D6 COMMA... RBDU SLOT (BE61)
A7DB GET P#2 NON-ZERO >A79E
A7DB MUST BE $5 THAN 8
A7DD AND LESS "RANGE ERROR" >>A79E
A7DF OR ELSE...YES (BE62)
A7DE CHECK >? EITHER 1 OR 2
A7F6 IS THIS >A7BF
A7F0 NO... IT'S A PROGRAM RUNNING? (BE42)
A7F7 YES, &A7BF
A7F5 >>A7D7 DIRECT COMMAND
A7F7 NO, "^/" DIRECT COMMAND
A7FA RETURNING NO PATHNAMES? >>A83D
A7FB EXPECTS BE55
A7FD NO... AND D VALID FOR THIS CMD?
A800 ARE S.83D
A802 NO >>A74S WE GOT PATHNAME? (BE56)
A804 YES, &A813
A808 YES >>
A80D IS PATHNAME REQUIRED?
A80F YES, "SYNTAX ERROR" >>A879
A811 NO, OPTIONAL - NO PREFIX YET?
A816 DOES PATHNAME START WITH: "/"?
A818 YES, FULLY QUALIFIED >>A81F
A81A NO, IS THERE A PREFIX ACTIVE?
A81D NO >>A838
A81F YES, (BE57)
A822 SLOT/DRIYE GIVEN WITH THIS CMD?
A824 NO, FORGET IT >>A83D
A826 YES, DO WE HAVE PATHNAME ALSO?
A828 NO,
A82A NULL OUT PATHNAME (BCBC)
A832 MARK THAT WE WILL HAVE ONE SO THEN >>A83D
A83B ADD PREFIX TO FILENAMES <A873>?
A83D ERROR? >>A87B
A83D GET COMMAND NUMBER (BE53) (BF9A)
A840 *2 AS INDEX INTO TABLE
A842 GET ADDRESS OF COMMAND HANDL
A84B AND STORE IT FOR INDIRECT JMP HAND?
A850 EXTERNAL COMMAND? IF SO GO NOW
A852 MY OWN COMMAND, "PREFIX"? >>A838
A854 YES, GO NOW >>A876
A855 S OR D VALID KEYWORDS FOR THIS
A858 PATHNAME GIVEN WITH THIS CMD?
A861 NO, GO NOW >>A876
A863 YES, GET FILE INFO FOR PATHNAME
A866 NO ERRORS I HOPE >>A876
A868 ERROR WAS PATH NOT FOUND? RG ROUTINE (B93F)
A86A NO, REAL ERROR - SAY SO >>A873(BCBC)
A86F CAN WE CREATE PATHNAME? 1! >>A876
A871 YES, OK THEN >>A876
A873 ELSE, "PATH NOT FOUND" CMD?
A875 RETURN THIS WITH PREFIX LENGTH (B201)
A876 MORE THAN 64 CHAR?
A876 IF SO, "SYNTAX ERROR" >>A927
A876 UPDATE LENGTH (B260)
A890 COPY PATHNAME(2) FORWARD TO MAKE ROOM (B281)
A891 PUT A "/" IN FIRST
A891 THEN THE PREFIX AND ANOTHER SLASH (B281)
A892 DONE!
A892 ***************** KEYWORD LOOKUP *******************
A892 ZERO THE ACCUMULATOR (CBA77)
A892 NINE POSSIBLE KEYWORDS IN TABLE
A892b COMPARE AGAINST EACH (B9BD)
A893b FOUND IT? >>A967
A893 NO, IS IT "T"? (FILE TYPE)
A8937 YES, OK THEN >>A93C
A893d ELSE, BAD KEYWORD >>A879
A893c IT'S "M", IS IT PERMITTED ON THIS CMD?
A894 NO, ERROR >>A963
A894 ELSE, MARK WE HAVE "T" (BE56)
A894 START WITH TYPE INDEX OF (BCD)
A895b INDICATE WHERE T VALUE IS TO GO (BCBC)
A8951b AND GO PARE ONE CHAR <A977>
A895d NOTING THERE???? >>A939
A896 IS IT A S?

Beneath Apple ProDOS Supplement

BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84 NEXT OBJECT ADDR: A8A0

---

ADDR  DESCRIPTION/CONTENTS

A800 ERROR? >>A87B
A805 DEFAULT DRIVE = PARSED DRIVE (B363)
A80B DEFAULT SLOT = PARSED SLOT (B363)
A80F PATHNAME1 STARTS WITH "/"?
A823 THEN ITS ALREADY GOT A PREFIX >>A926
A82B ELSE, GET LENGTH OF PATHNAME
A82A BUMP IT BY 2 (TO ALLOW FOR "/")
A82C WITH PREFIX WILL IT EXCEED 64 CHAR?
A82C YES, "SYNTAX ERROR" >>A927
A829 NO, UPDATE LENGTH TO INCLUDE PREFIX (BCBC)
A82F ---
A83D AND COPY PATHNAME1 FORWARD TO MAKE ROOM (BCBD)
A83C PUT A "/" AT THE BEGINNING
A83E AND AT THE END (BCBD)
A844 COPY PREFIX JUST READ TO START OF PATHNAME (B200)
A84A GET COMMAND NUMBER (BE53)
A84D "OPEN"
A84F YES, DONE NOW! >>A926
A861 "APPEND"?
A862 YES, DONE NOW! >>A926
A86C "EXEC"?
A86D YES, DONE NOW! >>A926
A86F ELSE, GET LENGTH OF PATHNAME2 (BO68)
A875b COMBINE THIS WITH PREFIX LENGTH (B201)
A890 MORE THAN 64 CHAR?
A896 IF SO, "SYNTAX ERROR" >>A927
A896 UPDATE LENGTH (B260)
A900 ---
A90F COPY PATHNAME2 FORWARD TO MAKE ROOM (B281)
A918 PUT A "/" IN FIRST
A91D THEN THE PREFIX AND ANOTHER SLASH (B281)
A926 ---
A927 DONE!

---

BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84 NEXT OBJECT ADDR: A8A0
BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84  NEXT OBJECT ADDR: 153

ADDR DESCRIPTION/CONTENTS

AA53 PATHNAME TOO LONG? (BCAA)
AA56, NO, CONTINUE COPYING >>AA40
AA58 ELSE, SET NOT-EQUAL CONDITION
AA5A AND EXIT

AA58 *********** COPY COMMAND NAME INTO TXTBUF ***********

AA5A SET INDICES
AA5F GET NEXT NON-BLANK <AA8A>
AA62 COPY TO TXTBUF (BCBD)
AA66 COMMA?
AA68 YES, DONE >>AA77
AA6A BLANK?
AA6C YES, DONE >>AA77
AA6E RETURN?
AA70 YES, DONE >>AA78
AA72 AT MAX LENGTH (8)? (BCAA)
AA75 NO, CONTINUE >>AA5F
AA77 ELSE, SET NOT-EQUAL CONDITION
AA79 AND EXIT

AA7A *********** FLUSH TO NON-BLANK ***********

AA7A IGNORE BLANKS
AA7F GET NEXT NON-BLANK <AA8A>
AA82 COMMA?
AA84 YES, OUT >>AA89
AA86 RETURN?
AA88 EXIT INDICATING WHAT WE FOUND
AA89 RETURN

AA8A *********** GET NEXT CHARACTER ***********

AA8A GET NEXT CHAR IN INPUT LINE (#200)
AA8D FORCE OFF MSB
AA8F LOWER CASE?
AA91 NO >>AA95
AA93 YES, FORCE UPPER CASE
AA95 BUMP LINE INDEX
AA96 IS THIS A FLUSH CHARACTER (LIKE BLANK)? (BCA9)
AA99 YES, GO GET NEXT ONE >>AA8A
AA9B ELSE, RETURN WITH IT

AA9C *********** CONVERT DIGIT AND ADD TO ACCUM ***********

AA9C NUMERIC?
AA9D NO >>AA4A
AA92 YES >>AA98
AA94 NOT NUMERIC, EXIT WITH CARRY SET
AA95 AND 2-FLAG RESET
AA97 RETURN
AA98 ISOLATE DECIMAL PORTION OF DIGIT
AA9A CURRENT VALUE OF ACCUM... (BCB1)
AA9E >1,763,936?
AA9B YES, OVERFLOW >>AA4A
AA9D PUSH ENTIRE ACCUM ONTO STACK (BCAF)
AA9B ACCUM*2 (RLT ONCE) <AB17>
AA9E ACCUM*4 (AND AGAIN) <AB17>
AA9C --
AA95 ACCUM OVERFLOW? >>AA6A
AA97 NO, ADD NEW DIGIT TO ACCUM (BCAF)
AA9A AND STORE IT (BCAF)
AA9D NO CARRY? >>AAED
AA9E GOT CARRY, PROPAGATE IT THRU ACCUM (BCB0)
AA9A OVERFLOW ERROR
AA9E NORMAL EXIT

AA9E *********** CONVERT HEX DIGIT AND ADD ***********

AA9E NUMERIC?
AA9F NO >>AAFE
AA94 YES >>AB04
AA96 NON-NUMERIC, HOW BOUT "$" THRU
AA9A "$"
AA9C YES! >>AB02
AA9F --
AA9F NO, GET OUT NOW
AB01 RETURN
AB02 "$" THRU "$", CONVERT TO SBA-SBF
AB04 ISOLATE DIGIT
AB06 SHIFT ACCUM 4 BITS LEFT TO MAKE ROOM <AB17>
AB08 (WATCH OUT FOR OVERFLOW) >>AA9A
AB10 OR IN NEW NIBBLE (BCAF)
AB11 AND REPLACE IN ACCUM LSB (BCAF)
AB16 DONE

AB17 *********** SHIFT 3 BYTE ACCUM LEFT A BIT ***********
**Beneath Appl.**

**BASIC Interpreter (SI) -- V1.0.1 -- 1 JAN 84 NEXT OBJECT ADDR: AB17**

**Addr Description/Contents**

**AB17** CLEAR ERROR FLAG
**AB18** CLEAR ERROR NUMBER
**AB19** CLEAR ERROR NUMBER ETC., MODE = 4 <AC19>
**AB20** JUMP INTO APPLESOFT TO RUN PROGRAM >D702

**AC19** SET NORMAL (NON-INVERSE OR FLASH) <F273>
**AC1A** SEARCH CHARACTER FOR TRACE IS "#" (9F98)
**AC23** NO COMMAND NUMBER NOW (BE53)
**AC26** NO PROMPT
**AC2A** SET MODE-4 (DEFERRED) <9FA0>
**AC2D** "SYNTAX ERROR" IF THINGS GO WRONG >>A879

**AB20** NO FILE ACTIVE NOW
**AB27** NO APPLESOFT ERROR NUMBER
**AB2C** GOSH PATHNAME?
**AB2D** NO, ERROR >>AC19
**AB2F** YES, LOAD PROGRAM <<AC42>
**AC20** ERROR? >>AC58
**AC24** NO, CLEAR VARIABLES <D665>

**AC27** CLEAR ERROR FLAG
**AC29** POSITION 70 LINE NUMBER IF GIVEN <<ACDC>
**AC2C** RESTORE MY INTERCEPTS <9AB0>
**AC2F** CLEAR COMMAND NUMBER ETC., MODE = 4 <AC19>

**AC31** CLEAR PIPELINE
**AC32** CLEAR PIPELINE ETC. <<AC19>
**AC37** CLEAR PIPELINE ETC. <<AC19>
**AC39** CLEAR PIPELINE ETC. <<AC19>
**AC3A** CLEAR PIPELINE ETC. <<AC19>
**AC3C** CLEAR PIPELINE ETC. <<AC19>
**AC3E** CLEAR PIPELINE ETC. <<AC19>

**AC46** SET COMMAND NUMBER ETC. ******** CLEAR COMMAND NUMBER ETC. ********

**AC49** ELSE, "FILE TYPE MISMATCH"
**AC53** RETURN

******** RUN "SYS" FILE ******

**AB5D** CLOSE ALL OPEN FILES <B54C>
**AB6A** CLOSE EXEC <B355>
**AB6B** LSB OF AS IS 00 (BE58)
**AB6C** FREE UP ALL OF B1'S MEMORY (BE68)
**AB6B** A2000 IS WHERE IT WILL LOAD (BE59)
**AB6C** TYPE IS "SYS" (BE6A)
**AB6D** FORCE, T, PATHNAME1, AD PARMS (BE56)
**AB6D** GO DO A STANDARD BRUN >>A85B

**AB6D** CLOSE ALL OPEN FILES <B54C>
**AB6A** CLOSE EXEC <B355>
**AB6B** LSB OF AS IS 00 (BE58)
**AB6C** FREE UP ALL OF B1'S MEMORY (BE68)
**AB6B** A2000 IS WHERE IT WILL LOAD (BE59)
**AB6C** TYPE IS "SYS" (BE6A)
**AB6D** FORCE, T, PATHNAME1, AD PARMS (BE56)
**AB6D** GO DO A STANDARD BRUN >>A85B

**AB6D** "CHAIN" COMMAND ********

**AB6D** SQUASH VARIABLES UP AGAINST HIMEM <<A449>
**AB65** SAVE HIMEM (BC78)
**AB6C** SET NEW HIMEM BELOW COMBINED VARS
**AB6D** LOAD FILE (LEAVE OTHERS OPEN) <<AC47>
**AB6E** RESTORE OLD HIMEM
**AB6F** ERROR? >>AC58
**AB78** NO, CLEAR VARIABLES <D665>
**AB7B** REEXPAND VARIABLES DOWN AGAINST LOMEM <<A4AF>

**AB7E** THEN OF NEW PROGRAM FUREST

**AB82** ********** "RUN" COMMAND **********

**AB82** NO INPUT FILE ACTIVE NOW
**AB87** NO APPLESOFT ERROR NUMBER
**AB8C** GOSH PATHNAME?
**AB8D** NO, ERROR >>AC19
**AB8F** YES, LOAD PROGRAM <<AC42>
**AC20** ERROR? >>AC58
**AC24** NO, CLEAR VARIABLES <D665>

**AC27** CLEAR ERROR FLAG
**AC29** POSITION 70 LINE NUMBER IF GIVEN <<ACDC>
**AC2C** RESTORE MY INTERCEPTS <9AB0>
**AC2F** CLEAR COMMAND NUMBER ETC., MODE = 4 <AC19>
**AC31** CLEAR PIPELINE
**AC32** CLEAR PIPELINE ETC. <<AC19>
**AC37** CLEAR PIPELINE ETC. <<AC19>
**AC39** CLEAR PIPELINE ETC. <<AC19>
**AC3A** CLEAR PIPELINE ETC. <<AC19>
**AC3C** CLEAR PIPELINE ETC. <<AC19>
**AC3E** CLEAR PIPELINE ETC. <<AC19>

**AC46** SET COMMAND NUMBER ETC. ******** CLEAR COMMAND NUMBER ETC. ********

**AC49** ELSE, "FILE TYPE MISMATCH"
**AC53** RETURN

******** RUN "SYS" FILE ******

**AB5D** CLOSE ALL OPEN FILES <B54C>
**AB6A** CLOSE EXEC <B355>
**AB6B** LSB OF AS IS 00 (BE58)
**AB6C** FREE UP ALL OF B1'S MEMORY (BE68)
**AB6B** A2000 IS WHERE IT WILL LOAD (BE59)
**AB6C** TYPE IS "SYS" (BE6A)
**AB6D** FORCE, T, PATHNAME1, AD PARMS (BE56)
**AB6D** GO DO A STANDARD BRUN >>A85B

**AB6D** "CHAIN" COMMAND ********

**AB6D** SQUASH VARIABLES UP AGAINST HIMEM <<A449>
**AB65** SAVE HIMEM (BC78)
**AB6C** SET NEW HIMEM BELOW COMBINED VARS
**AB6D** LOAD FILE (LEAVE OTHERS OPEN) <<AC47>
**AB6E** RESTORE OLD HIMEM
**AB6F** ERROR? >>AC58
**AB78** NO, CLEAR VARIABLES <D665>
**AB7B** REEXPAND VARIABLES DOWN AGAINST LOMEM <<A4AF>

**AB7E** THEN OF NEW PROGRAM FUREST

**AB82** ********** "RUN" COMMAND **********

**AB82** NO INPUT FILE ACTIVE NOW
**AB87** NO APPLESOFT ERROR NUMBER
**AB8C** GOSH PATHNAME?
**AB8D** NO, ERROR >>AC19
**AB8F** YES, LOAD PROGRAM <<AC42>
**AC20** ERROR? >>AC58
**AC24** NO, CLEAR VARIABLES <D665>

**AC27** CLEAR ERROR FLAG
**AC29** POSITION 70 LINE NUMBER IF GIVEN <<ACDC>
**AC2C** RESTORE MY INTERCEPTS <9AB0>
**AC2F** CLEAR COMMAND NUMBER ETC., MODE = 4 <AC19>
**AC31** CLEAR PIPELINE
**AC32** CLEAR PIPELINE ETC. <<AC19>
**AC37** CLEAR PIPELINE ETC. <<AC19>
**AC39** CLEAR PIPELINE ETC. <<AC19>
**AC3A** CLEAR PIPELINE ETC. <<AC19>
**AC3C** CLEAR PIPELINE ETC. <<AC19>
**AC3E** CLEAR PIPELINE ETC. <<AC19>

**AC46** SET COMMAND NUMBER ETC. ******** CLEAR COMMAND NUMBER ETC. ********

**AC49** ELSE, "FILE TYPE MISMATCH"
**AC53** RETURN

******** RUN "SYS" FILE ******

**AB5D** CLOSE ALL OPEN FILES <B54C>
**AB6A** CLOSE EXEC <B355>
**AB6B** LSB OF AS IS 00 (BE58)
**AB6C** FREE UP ALL OF B1'S MEMORY (BE68)
**AB6B** A2000 IS WHERE IT WILL LOAD (BE59)
**AB6C** TYPE IS "SYS" (BE6A)
**AB6D** FORCE, T, PATHNAME1, AD PARMS (BE56)
**AB6D** GO DO A STANDARD BRUN >>A85B

**AB6D** "CHAIN" COMMAND ********

**AB6D** SQUASH VARIABLES UP AGAINST HIMEM <<A449>
**AB65** SAVE HIMEM (BC78)
**AB6C** SET NEW HIMEM BELOW COMBINED VARS
**AB6D** LOAD FILE (LEAVE OTHERS OPEN) <<AC47>
**AB6E** RESTORE OLD HIMEM
**AB6F** ERROR? >>AC58
**AB78** NO, CLEAR VARIABLES <D665>
**AB7B** REEXPAND VARIABLES DOWN AGAINST LOMEM <<A4AF>

**AB7E** THEN OF NEW PROGRAM FUREST
BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84  NEXT OBJECT ADDR: AC20
ADDR DESCRIPTION/CONTENTS

AC30  *********** "LOAD" COMMAND  ***********************
AC30  LOAD PROGRAM <AC42>
AC33  ERROR? IF NOT, FALL THRU TO WARMSTART >>AC58

AC35  *********** WARMDOs: WARMSTART BI ****************
AC35  CLEAR APPLESOFT, RESET POINTERS <DE65>
AC38  RESET MODE/SET INTERCEPTS <9A17>
AC3D  CURSOR HORIZ. = 0 (START OF LINE)
AC3F  GO WARMSTART APPLESOFT >>D43F

AC42  *********** LOAD A PROGRAM  ***********************
AC42  CLOSE ALL OPEN FILES <B54C>
AC45  ERROR? >>AC58
AC47  GO LOAD FILE <AC59>
AC4A  ERROR? >>AC58
AC4C  SET LOMEM = ARRAYS = FREESTART
AC4E  ALL TO END OF PROGRAM LOADED
AC58  RETURN

AC59  *********** READ A PROGRAM FROM A FILE  **********
AC59  READ REQUESTED
AC5B  TYPE = BAS ASSUMED
AC5D  OPEN THE FILE <B1EE>
AC60  ERROR? >>AC58
AC64  MLI: GET EOF <BE70>
AC67  ERROR? >>AC58
AC6B  APPLESOFT PROGRAM START --> READ DATA (BED7)
AC6E  ADD TO THAT THE EOF MARK TO ... (BEC8)
AC71  SET AD PARM --> END OF PROGRAM IMAGE (BE58)
AC7F  OVERFLOW? >>AC83
AC81  NO, WOULD PROGRAM EXCEED HIMEM?
AC83  IF SO...
AC85  "PROGRAM TOO LARGE" >>AC58
AC87  ELSE, PICK UP LENGTH AGAIN (BEC8)
AC90  AND GO READ IT IN <BBBB>
AC90  ERROR? >>AC58
AC92  CLOSE FILE <AFFC>
AC95  ERROR? >>AC58
AC97  RELOCATE PROGRAM IF NECESSARY <AC5A>
AC9A  COPY AD PARM TO APPLESOFT PGM END PTR
AC94  RETURN

ACA4  *********** RELOCATE APPLESOFT PROGRAM  ************
ACCA  *********** POSITION TO LINE NUMBER  *************
ACDC  WAS A LINE NUMBER PARM GIVEN? (BEB7)
ACD2  NO, NEVER MIND >>ACD9
ACD4  COPY L KEYWORD VALUE TO APPLESOFT'S LINE # (BE68)
ACD8  THEN CALL APPLESOFT TO FIND THE LINE <D61A>
ACF4  SUBTRACT ONE FROM THE ADDRESS
ACF6  AND POINT APPLESOFT'S GETCHR SUBROUTINE
ACF8  AT IT (SO NEXT CHAR READ WILL BE FIRST
ACFA  CHARACTER ON THE LINE).
ACFB  RETURN

AD00  *********** "SAVE" COMMAND  *********************
AD00  DOES FILE EXIST ALREADY? >>AD24
AD02  NO, TYPE = BAS
AD04  IN T KEYWORD VALUE (BE6A)
AD07  AND MLI LIST (BEB8)
AD0B  ALLOW ALL ACCESSES (READ/WRITE/ETC.) (BEB7)
AD11  SAVE PROGRAM START ADDRESS IN (BEA5)
AD14  AUXID'S (BEB9)
AD1F  GO CREATE A NEW FILE <ADBB>
AD22  ERROR? >>AD60
AD24  WRITE ACCESS REQUESTED
AD26  BAS TYPE FILE
AD28  OPEN IT <B1EE>
AD2B  ERROR? >>AD60
AD30  SUBTRACT APPLESOFT PTRS TO COMPUTE
AD32  LENGTH OF PROGRAM.
AD33  STORE THIS IN EOF MARK LIST (BEC8)
AD48  MSB OF EOF MARK IS 00 (64K PGM) (BC8A)
AD45  POINT LIST TO PROGRAM AS DATA TO WRITE (BED7)
AD4D  WRITE A RANGE TO DISK FILE <BB04>
AD50  ERROR? >>AD60
AD54  MLI: SET EOF (TO TRUNCATE OLD LONGER FILE) <BE7B>
AD57  ERROR? >>AD60
AD59  CLOSE THE FILE <AFFC>
AD5C  ERROR? >>AD60
BASIC Interpreter (B1) -- V1.0.1 -- 1 JAN 84     NEXT OBJECT ADDR: AD66

ADDR     DESCRIPTION/CONTENTS
AD60     DOES PROGRAM START MATCH AUXID IN FILE INFO?
AD65     NO, CHANGE IT >>AD66
AD66     ELSE, EXIT
AD68     TO CHANGE IT, (BE89)
AD74     EXIT THRU SET FILE INFO ROUTINE >>BE83

AD77     ************ "CREATE" COMMAND **********************
AD77     AUXID = 0 (AS OR RECLN)
AD82     TYPE KEYWORD GIVEN?
AD84     YES >>AD88
AD88     NO, ASSUME TYPE = DIR (BE6A)
AD88     *** CREATE FILE ENTRY *** (BE43)
AD8E     EXEC FILE ACTIVE?
AD91     HOW MANY FILES ARE OPEN INCLUDING EXEC? (BE4D)
AD94     8 OR MORE?
AD96     YES, ERROR >>AD93
AD9B     ELSE, SET TYPE IN MLI LIST (BEA4)
AD9E     FULL ACCESS (READ/WRITE/ETC.)
ADA0     KIND = STANDARD FILE
ADA2     DIR FILE WANTED?
ADA4     NO >>ADA8
ADA6     YES, KIND = DIR FILE
ADA8     SET ACCESS (BEA3)
ADAB     AND KIND (BEA7)
ADB0     MLI: CREATE (DON'T COME BACK HERE) >>BE70
ADB3     "RAM TOO LARGE" ERROR
AD55     RETURN

ADB6     ************ "RENAME" COMMAND **********************
ADB6     --
ADBA     SECOND PATHNAME GIVEN?
ADBD     IF SO, GO MLI: RENAME >>ADC4
ADBF     "SYNTAX ERROR" OTHERWISE >>AB79

ADC2     ************ "DELETE" COMMAND **********************
ADC2     SETUP MLI: DELETE CALL TYPE
ADC4     EXIT THRU MLI CALL >>BE70

ADC7     ************ "LOCK" COMMAND **********************
ADC7     GET FILE INFO FOR PATHNAME1 <BE8A>
ADCA     GET ACCESS CODES (BE7)
ADCB     TURN OFF ALL...
ADCF     BUT READ
ADD4     THEN GO SET UPDATED FILE INFO >>BE81
<table>
<thead>
<tr>
<th>BASIC Interpreter (B1) -- V1.0.1 -- 1 JAN 84</th>
<th>BASIC Interpreter (B1) -- V1.0.1 -- 1 JAN 84</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADDR DESCRIPTION/CONTENTS</td>
<td>ADDR DESCRIPTION/CONTENTS</td>
</tr>
<tr>
<td>AE55 &quot;PATH NOT FOUND&quot; ERROR</td>
<td>AECC ---</td>
</tr>
<tr>
<td>AE59 ---</td>
<td>AECA MLI: GET EOF &lt;BE70&gt;</td>
</tr>
<tr>
<td>AE5A RETURN</td>
<td>AECD ERROR? &gt;&gt;AEDB</td>
</tr>
<tr>
<td>AE5B &quot;BLOAD&quot; COMMAND</td>
<td>AECE GET L (EOF MARK) (BEC8)</td>
</tr>
<tr>
<td>(DOES NOT SET MODE=4 SO DOS COMMANDS MAY</td>
<td>AEDE BETTER NOT EXCEED 64K (Beca)</td>
</tr>
<tr>
<td>NOT BE ISSUED AS WITH A BASIC PROGRAM)</td>
<td>AEED NO.. &gt;&gt;AEED</td>
</tr>
<tr>
<td>AE5B BLOAD IT FIRST &lt;AE68&gt;</td>
<td>AE79 YES, &quot;PROGRAM TOO LARGE&quot;</td>
</tr>
<tr>
<td>AE5E ERROR? &gt;&gt;AE59</td>
<td>AE80 ---</td>
</tr>
<tr>
<td>AE60 THEN CALL IT &lt;AE65&gt;</td>
<td>AE81 RETURN</td>
</tr>
<tr>
<td>AE63 THEN EXIT</td>
<td>AE82 STORE LENGTH TO READ OR WRITE (BED9)</td>
</tr>
<tr>
<td>AE64 RETURN</td>
<td>AE83 B KEYWORD GIVEN?</td>
</tr>
<tr>
<td>AE65 INDIRECT JMP TO BINARY PROGRAM &gt;&gt;BED7</td>
<td>AE84 NO &gt;&gt;AEF0</td>
</tr>
<tr>
<td>AE68 &quot;BLOAD&quot; COMMAND</td>
<td>AE85 YES, COPY B VALUE TO SET MARK LIST</td>
</tr>
<tr>
<td>AE68 READING...</td>
<td>(BE5A)</td>
</tr>
<tr>
<td>AE6A TYPE = BIN</td>
<td>AEF5 ---</td>
</tr>
<tr>
<td>AE6B OPEN THE FILE &lt;BIEE&gt;</td>
<td>AEF7 MLI: SET MARK &lt;BE70&gt;</td>
</tr>
<tr>
<td>AE6F ERROR? &gt;&gt;AE59</td>
<td>AEF8 NO ERROR? &gt;&gt;AEF0</td>
</tr>
<tr>
<td>AE70 ASSUME USER SPECIFIED AD KEYWORD</td>
<td>AEF9 ERROR, RANGE ERROR?</td>
</tr>
<tr>
<td>(BE58)</td>
<td>AEFF NO &gt;&gt;AEFF</td>
</tr>
<tr>
<td>AE7A IF SO, USE HIS ADDRESS &gt;&gt;AEBC</td>
<td>AEF1 MLI: BSAVEVING (NOT BLOAD/BRUNING)?</td>
</tr>
<tr>
<td>AE7C ELSE, USE AD IN FILE INFO AUXID (BE9B)</td>
<td>AEF2 MLI: SET EOF FORWARD TO MARK &lt;BE70&gt;</td>
</tr>
<tr>
<td>AE7C WAS T KEYWORD GIVEN?</td>
<td>AEF3 AND TRY SET MARK AGAIN &gt;&gt;AEF5</td>
</tr>
<tr>
<td>AE7C YES, INVALID PARM (ONLY BIN IS LEGAL)</td>
<td>AEF4 RETURN</td>
</tr>
<tr>
<td>&gt;&gt;AEF5</td>
<td>AEF5 GET COMMAND NUMBER (BE53)</td>
</tr>
<tr>
<td>AE7C POINT READ/WRITE PARM TO DATA (BED7)</td>
<td>AE12 ASSUME READ</td>
</tr>
<tr>
<td>AE92 AND SAVE THIS ADDRESS IN AUXID (BE9B)</td>
<td>AE13 BSAVEV</td>
</tr>
<tr>
<td>AE98 PICK UP LENGTH FROM L KEYWORD VALUE</td>
<td>AE14 ELSE, BSAVING... (BE57)</td>
</tr>
<tr>
<td>(BE5F)</td>
<td>AE15 NO, READ IS CORRECT &gt;&gt;AEF2</td>
</tr>
<tr>
<td>AE9E WAS L OR E GIVEN?</td>
<td>AE16 ELSE, BSAVEV... (BE57)</td>
</tr>
<tr>
<td>AEAE NO &gt;&gt;AECC</td>
<td>AE17 MLI: RESAVING, GO RIGHT NOW &gt;&gt;AEF3</td>
</tr>
<tr>
<td>AE2A BOTH?</td>
<td>AE18 MLI: UPDATE EOF TO NEW PLACE (BEC8)</td>
</tr>
<tr>
<td>AE4A YES... &quot;NAUGHTY!&quot; &gt;&gt;AECC</td>
<td>AE19 MLI: SET EOF &lt;BE70&gt;</td>
</tr>
<tr>
<td>AE6A E GIVEN?</td>
<td>AE20 MLI: WRITING</td>
</tr>
<tr>
<td>AE68 NO, MUST BE L &gt;&gt;AEED</td>
<td>AE21 MLI: READ OR WRITE &lt;BE70&gt;</td>
</tr>
<tr>
<td>AE7A YES... (BE50)</td>
<td>AE22 ERROR? &gt;&gt;AEAB</td>
</tr>
<tr>
<td>AE78 COMPUTE L = (E - AD) (BE58)</td>
<td>AE23 NO, BSAVEV</td>
</tr>
<tr>
<td>AE7A PLUS ONE FOR INCLUSIVE RANGE &gt;&gt;AEBD</td>
<td>AE24 MLI: READ OR WRITE &lt;BE70&gt;</td>
</tr>
<tr>
<td>AE7B MAKE SURE NO BORROW OCCURED &gt;&gt;AEED</td>
<td>AE25 ERROR? &gt;&gt;AEAB</td>
</tr>
<tr>
<td>AE7C OR ELSE, &quot;RANGE ERROR&quot;</td>
<td>AE26 NO &gt;&gt;AEF1</td>
</tr>
<tr>
<td>AECC &quot;INVALID PARM&quot; ERROR</td>
<td>AE27 MLI, CREATE THE FILE &lt;ADB8&gt;</td>
</tr>
<tr>
<td>AE6A RETURN</td>
<td>AE28 THEN EXIT THRU CLOSE &gt;&gt;AEFC</td>
</tr>
<tr>
<td>AE6B &quot;BLOAD&quot; COMMAND</td>
<td>AE41 MLI: &quot;STORE&quot; COMMAND ***</td>
</tr>
<tr>
<td>AE7A IF SO, USE HIS ADDRESS &gt;&gt;AEBC</td>
<td>AE55 &quot;COMPRESS APPLESOFT VARS AGAINST HIMEM (A449)&quot;</td>
</tr>
</tbody>
</table>
Beneath Apple ProDOS Supplement

BASIC Interpreter (RI) -- V1.0.1 -- 1 JAN 84 NEXT OBJECT ADDR: AF5C

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CURRENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>AF5C</td>
<td>OPEN &quot;VAR&quot; FILE FOR WRITE (&lt;BEE&gt;)</td>
</tr>
<tr>
<td>AF5F</td>
<td>ERROR? &gt;&gt;AF9A</td>
</tr>
<tr>
<td>AF61</td>
<td>POINT TO INTERNAL 5 BYTE HEADER BUFFER &lt;AFA2&gt;</td>
</tr>
<tr>
<td>AF64</td>
<td>AND WRITE OUT LENGTHS OF VARS &lt;B004&gt;</td>
</tr>
<tr>
<td>AF67</td>
<td>ERROR? &gt;&gt;AF9A</td>
</tr>
<tr>
<td>AF69</td>
<td>STORE ADDRESS OF VARS (BC8E)</td>
</tr>
<tr>
<td>AF6C</td>
<td>IN READ/WRITE PARM LIST: (BED7)</td>
</tr>
<tr>
<td>AF6P</td>
<td>AND FILE INFO AUXID (BB9)</td>
</tr>
<tr>
<td>AF7B</td>
<td>GET LENGTH OF VARS (BC91)</td>
</tr>
<tr>
<td>AF81</td>
<td>AND WRITE THEM OUT &lt;B004&gt;</td>
</tr>
<tr>
<td>AF84</td>
<td>ERROR? &gt;&gt;AF9A</td>
</tr>
<tr>
<td>AF88</td>
<td>MLI: SET NEW EOF (TRUNCATE IF NECESSARY) &lt;BE70&gt;</td>
</tr>
<tr>
<td>AF90</td>
<td>ERROR? &gt;&gt;AF9A</td>
</tr>
<tr>
<td>AF92</td>
<td>SET FILE INFO WITH AD OF VARS &lt;B833&gt;</td>
</tr>
<tr>
<td>AF95</td>
<td>ERROR? &gt;&gt;AF9A</td>
</tr>
<tr>
<td>AF97</td>
<td>CLOSE FILE &lt;AFC&gt;</td>
</tr>
<tr>
<td>AF9A</td>
<td>---</td>
</tr>
<tr>
<td>AF9C</td>
<td>REEXPAND VARS BACK AGAIN &lt;A4AF&gt;</td>
</tr>
<tr>
<td>AFA1</td>
<td>RETURN</td>
</tr>
</tbody>
</table>

AFA2 ********** SETUP TO READ/WRITE VAR HDR **************
APPLESOFT VARIABLES HEADER CONSISTS OF:
2 BYTE LENGTH OF SIMPLE ARRAY VARIABLES
2 BYTE LENGTH OF SIMPLE VARIABLES ONLY
1 BYTE MSB OF HIMEM FOR THESE VARIABLES

AFA2 STORE ADDRESS OF 5 BYTE INFO
AFA4 IN READ/WRITE PARM LIST (BED7)
AFAE LENGTH = 5
AF00 RETURN

APB1 ********** "RESTORE" COMMAND **************

APB1 TYPE = VAR
APB3 PRINTING
APB5 OPEN THE FILE <BEE>
APB8 ERROR? >>AFA1
APB9 SET UP TO READ THE HEADER <AFA2>
APBD READ 5 BYTE HEADER <B004>
APC0 ERROR? >>AFA1
APC2 PICK UP WHERE TO READ IN COMPRESSED VARS (BEB9)
APC5 FROM AUXID (BCE8)
APC8 ADJUST MSB OR THIS BY THE DIFFERENCE
APC8 BETWEEN HIMEM'S (NOW AND WHEN STORED) (BC8D)
APDB MAKE SURE VARS WON'T OVERLAY PROGRAM
APDD IF SO, ERROR >>APF8
APF7 COMPUTE LENGTH OF ALL VARS/STRINGS
APF9 (HIMEM-START) (BC8E)
APF0 ERROR? >>AFA1

BASIC Interpreter -- KSWL (BCE9)
ADDRESS L-value with (0036)

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CURRENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>AF02</td>
<td>CLOSE THE FILE: WORD VALUE &lt;B061&gt;</td>
</tr>
<tr>
<td>AF05</td>
<td>EXIT BY REPLY</td>
</tr>
<tr>
<td>AF08</td>
<td>&quot;PROGRAM TERMINATING&quot; GOST 1/0 NOW</td>
</tr>
<tr>
<td>AF0B</td>
<td>RETURN</td>
</tr>
</tbody>
</table>

AFPC ************ |
AFPC SET MLI CLASSE FILE
AFEE AND GO TO |

B000 ************ RGL >>RGBC |
B000 READ MLI OPCODE |
B002 JUMP IN >> |
B004 WRITE MLI OPCODE |
B006 STORE LENGTH >> |
B00C EXIT THRU OPCODE |

B00F ************ "MLI READ-" |
B00F USE KSWL ADDR: COMM0 |
B014 JUMP TO COMM0 |

B016 ************ "MLI OUTPUT-" |
B016 USE KSWL IN: COMM0 |
B01B AND INVEC |

B01D OR IN SLOT |
B020 *2 FOR USE |
B025 WAS SLOT P GIVEN BY END VARS DOWN >>AF9A |
B027 NO... >>B89/AS INDEX |
B029 YES, (BE78X/LOAD) >>B00 |
B02C AD GIVEN? |
B02E NO, GET IN |
B031 AND STORE >>B94 |
B03A VALIDITY CVOL OR OUT |
B03D NO GOOD? >>AD KEY |
B03F GET INDEX PCK 1/O-5 |
B045 AND REPLACE FILE |
B048 HIS ADDRESS AS SWL OR VARS |
B04E RETURN & ONE OF IT |
B04F VALIDITY CVOL |
B052 NO GOOD? >> |
B054 GOOD, COPY FROM AD KE |
B060 EXIT BUT B060 >>WHITE >>BE70
B061  ***********  VALIDITY CHECK I/O DRIVER  ***************

B061  $3A/3B -- > NEW HANDLER (FROM AD PARM) (BB58)
B060  IS DRIVER IN MAIN RAM (BELOW $C000)?
B06F  YES >>B066
B071  NO, RESUME I/O CARD RAMS (CFFF)
B074  USE 23C TO COUNT ITERATIONS
B076  TEST ROM AT USER'S ADDRESS
B07C  FOR STABILITY
B080  256 TIMES
B084  MUST BE OK
B085  RETURN
B086  MAIN RAM I/O DRIVER
B088  MUST START WITH A "CLD" INSTRUCTION
B08A  OK... >>B04
B08C  ELSE, "NO DEVICE CONNECTED"
B08F  RETURN

B090  ***********  "CAT" COMMAND  ***********************

B090  39 CHARACTERS PER LINE
B092  THEN PROCESS LIKE "CATALOG" >>B096

B094  ***********  "CATALOG" COMMAND  *********************

B094  79 CHARACTERS PER LINE
B096  STORE LINE LENGTH (BC86)
B09C  GET T FOR T AND
B09E  ...PATHNAME GIVEN
B09F  GOT T >>B0A4
B0A1  NO T, T=0 (ANY TYPE WILL DO) (BE6A)
B0A4  GOT PATHNAME >>BBAB
B0A6  NO PATHNAME, GET FILE INFO FOR PREFIX <B0A2>
B0A9  ERROR? >>B111
B0AB  OPEN/READ DIRECTORY HEADER <B1A4>
B0AE  ERROR? >>B111
B0B0  SKIP TO A NEW LINE <9FE2>
B0B3  FORMAT DIRECTORY'S NAME TO $201 <B12>
B0B6  PRINT $201 <9FD4>
B0B9  SKIP TO A NEW LINE <9FE2>
B0BC  BLANK $201 BUFFER <A6A9>
B0C1  UNPACK HEADLINE MESSAGE LINE <9FE7>
B0C4  PRINT IT (40 OR 80 COLUMNS) <9FD4>
B0C7  SKIP TO A NEW LINE <9FE2>
B0CD  ANY FILES IN THIS DIRECTORY? (BCBA)
B0DF  NO >>B0FD
B0D2  YES, READ NEXT ENTRY <B22B>
B0DF  ERROR? >>B111
B0D7  GET TYPE REQUESTED FOR SEARCH (BE6A)
B22B *************** READ NEXT DIRECTORY ENTRY ***************

B22B FORCE MARK TO START OF THIS BLOCK (BEC9)
B22B CHECK ENTRY NUMBER (BCBB)
B22B LAST ENTRY IN THIS BLOCK? (BCBB)
B22B NO >> B247
B22B YES, ENTRY 0 NEXT TIME (BCBB)
B22B BUMP MARK TO NEXT BLOCK (BEC9)
B22B ---
B22B MARK POSITIONED TO PROPER ENTRY YET? >> B252
B22B NO, BUMP POINTER TO NEXT ENTRY (BCB7)
B22B AND CONTINUE IF STILL FIRST PAGE >> B247
B22B JUST ENTERED SECOND PAGE >> B244
B22B ADD 4 TO PTR TO ADJUST FOR BLOCK PREFIX
B22B MLI; SET MARK >> B79
B22B ERROR? >> B77
B22B MLI; READ >> B79
B22B OPEN FILE >> B79
B22B MLI; READ >> B79
B22B BUMP ENTRY COUNTER (BCBB)
B22B IS THIS ENTRY VALID?
B22B NO, SKIP OVER IT >> B22B
B22B DECREMENT FILE COUNT (BCB9)
B22B AND RETURN TO CALLER

B278 *************** EXTERNAL COMMAND HANDLER ***************

B278 INDIRECT JMP TO XTRNAD VECTOR >> B50

B27B *************** "EXEC" COMMAND ***************

B27B IS THIS FILE OPEN ALREADY? <479>
B27B NO >> B2A
B27B YES, EXEC CLOSING? (BE4E)
B27B NO >> B2A
B27B SAVE REFNUM (BEC7)
B27B RESET MARK TO ZERO (BEC8)
B27B MLI; SET MARK >> B79
B27B ERROR? >> B29F
B27B GET REFNUM AGAIN (BEC7)

B1A4 *************
B1A4 READ ONL
B1A9 CHECK FILE
B1AB VOLUME D
B1AD NO >> B1B9 XAD DIRECTORY HDR *************

B1B7 BUFFER I
B1C3 LENGTH I
B1CD MLI OPEN (B79)
B1CF ERROR? >> B1F
B1D4 COPY ENTRY >> B137F
B1D7 AND FILE
B1DD STORE ENTRY LENGTH
B1E2 SET COUNT ENTRY F
B1E7 MARK = ???
B1ED RETURN PET TO F;

B1EE ************
B1F2 T KEYWORD
B1F4 NO >> B1F
B1F6 YES, USING FILE
B1F9 ---
B1FA EXISTING KEYWORD
B1FB NO, ERR
B1FF CHECK ACTIVE FILE
B202 REQUESTED >> B22
B204 SET SYSTEM ACCESS
B20C LEVEL TO ACCESS

B211 MLI; OPEN (B79)
B214 ERROR? >> B222
B219 SAVE REFNUM IN READ/WRITE PARMLIST (BEC6)
B21C AND CLOSE PARMLIST (BEC6)
B21F AND GET/SET EOF/MARK LIST (BEC7)
B222 AND EXIT

B223 "FILE TYPE MISMATCH"
B226 RETURN
B227 "FILE LOCKED"
B22A RETURN

B164 STORE "/
B167 GET FILE
B16A ERROR?
B16C BLANK "$"
B171 UNKNOWN "$"
B174 ZERO THE BUFFER
B177 CONVERT BLOCKS TO CHARACTER (BCBD)
B17F CONVERT NBR TO "A" BLOCKS USED...
B19A BLOCKS FREE = (T) <6A9>
B19F CONVERT NBR TO "A" BLOCKS USED...
B1A3 DONE (BLOCKS FREE) = ACCUM <AB77>

B1A4 READ ONL
B1A9 CHECK FILE
B1AB VOLUME D
B1AD NO >> B1B9 XAD DIRECTORY HDR *************

B1B7 BUFFER I
B1C3 LENGTH I
B1CD MLI: READ >> B16D
B1DF ERROR? >> B1F
B1D4 COPY ENTRY >> B137F
B1D7 AND FILE
B1DD STORE ENTRY LENGTH
B1E2 SET COUNT ENTRY F
B1E7 MARK = ???
B1ED RETURN PET TO F;

B1EE ************
B1F2 T KEYWORD
B1F4 NO >> B1F
B1F6 YES, USING FILE
B1F9 ---
B1FA EXISTING KEYWORD
B1FB NO, ERR
B1FF CHECK ACTIVE FILE
B202 REQUESTED >> B22
B204 SET SYSTEM ACCESS
B20C LEVEL TO ACCESS

B211 MLI; OPEN (B79)
B214 ERROR? >> B222
B219 SAVE REFNUM IN READ/WRITE PARMLIST (BEC6)
B21C AND CLOSE PARMLIST (BEC6)
B21F AND GET/SET EOF/MARK LIST (BEC7)
B222 AND EXIT

B223 "FILE TYPE MISMATCH"
B226 RETURN
B227 "FILE LOCKED"
B22A RETURN

B22B *************** READ NEXT DIRECTORY ENTRY ***************

B22B FORCE MARK TO START OF THIS BLOCK (BEC9)
B233 CHECK ENTRY NUMBER (BCBB)
B23B LAST ENTRY IN THIS BLOCK? (BCBB)
B23B NO >> B247
B23E YES, ENTRY 0 NEXT TIME (BCBB)
B241 BUMP MARK TO NEXT BLOCK (BEC9)
B247 ---
B249 MARK POSITIONED TO PROPER ENTRY YET? >> B252
B24B NO, BUMP POINTER TO NEXT ENTRY (BCB7)
B24E AND CONTINUE IF STILL FIRST PAGE >> B247
B250 JUST ENTERED SECOND PAGE >> B244
B252 ADD 4 TO PTR TO ADJUST FOR BLOCK PREFIX
B259 MLI; SET MARK >> B79
B25C ERROR? >> B277
B25F MLI; READ >> B79
B263 ERROR? >> B277
B265 BUMP ENTRY COUNTER (BCBB)
B26B IS THIS ENTRY VALID?
B26D NO, SKIP OVER IT >> B22B
B26F DECREMENT FILE COUNT (BCB9)
B277 AND RETURN TO CALLER

B278 *************** EXTERNAL COMMAND HANDLER ***************

B278 INDIRECT JMP TO XTRNAD VECTOR >> B50

B27B *************** "EXEC" COMMAND ***************

B27B IS THIS FILE OPEN ALREADY? <479>
B27B NO >> B2A
B27B YES, EXEC CLOSING? (BE4E)
B283 NO >> B2A
B285 SAVE REFNUM (BEC7)
B28A RESET MARK TO ZERO (BEC8)
B285 MLI; SET MARK >> B79
B298 ERROR? >> B29F
B29A GET REFNUM AGAIN (BEC7)
**BASIC Interpreter** (B1) -- V1.0.1 -- 1 JAN 84  NEXT OBJECT ADDR: B29D

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>B29D</td>
<td>GO RESTART THIS EXEC FILE FROM ITS START &gt;&gt;B31D</td>
</tr>
</tbody>
</table>

```
******* CLOSE EXEC FILE ***********
```

```
B2FA PRESEVERE CALLEER'S AREG
B2AO AND CLOSE THE FILE <B355>
B2A5 THEN RETURN WITH ERROR
B2A6 "FILE BUSY" ERROR
B2A9 RETURN

******* CONTINUE EXEC SETUP ***********
```

```
B2AA EXEC ACTIVE? (BE43)
B2AD NO >>B2B4
B2AF YES, CLOSE IT <B355>
B2B4 GET FILE TYPE (BEB8)
B2B7 SHOULBE BE TXT
B2BD IT IS >>B2BF
```

```
B2BB ELSE, "FILE TYPE MISMATCH"
B2BD RETURN WITH ERROR
B2BE RETURN
```

```
B2BF MOVE STRINGS TO MAKE ROOM FOR A BUFFER <A232>
B2C2 NO ROOM? >>B2BD
B2C6 STORE NEW BUFFER ADDRESS IN PARM LIST (BEC9)
B2CF GET COUNT OF OPEN FILES (BE4D)
B2D2 NO OTHERS CURRENTLY OPEN? >>B2F8

******* MAKE EXEC TOPMOST BUFFER ******
```

```
B2D4 OTHERS ARE OPEN...
B2D6 OPENCOUNT*4 (4 PAGES PER BUFFER)
B2D8 ADD THIS TO MY BUFFER TO FIND TOP BUFFER (BC88)
B2DC SEARCH OPEN FILES TO FIND THE FILE WHICH (BC93)
B2DF IS USING THIS BUFFER... >>B2B5
B2E4 IF IT IS NOT FOUND, BREAK!
B2E5 --
B2E6 MOVE THAT FILE TO THE NEW BUFFER INSTEAD (BC93)
B2E9 GET THAT FILE'S REPNUM ALSO (BC9B)
```

```
B2F1 MLI SET BUFF <BE70>
B2F4 NO ERRORS? >>B2F7
B2F6 IF ERROR, BREAK!
B2F7 --
```

**BASIC Interpreter** (B1) -- V1.0.1 -- 1 JAN 84  NEXT OBJECT ADDR: B2F7

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

```
******** OPEN NEW EXEC FILE ***********
```

```
B2F8 SET NEW BUFFER ALLOCATION PAGE (BC88)
B2FB SET UP OPEN LIST FOR EXEC TOO (BECF)
B300 LEVEL = 0 (B9F4)
B305 MLI: OPEN (EXEC FILE) <BE70>
B309 NO ERROR? >>B311
B30A --
B30B IF ERROR, FREE BUFFER FIRST <A289>
B310 THEN EXIT WITH ERROR
B311 SAVE BUFFNO FOR EXEC (BECF)
B317 AND REPNUM TOO (BED0)

******* COMPLETE EXEC COMMAND ***********
```

```
B31D SAVE READ REPNUM (BED6)
B329 AND GET/SET REPNUM (BEC7)
B32B AND NEWLINE REPNUM (BED2)
B32F SET "L" VALUE FROM AUXID (BE5F)
B332 SAVE PATHNAME/AUXID IN OPEN FILE TABLE <B445>
B337 IGNORE MSB FOR END OF LINE CHAR (BED3)
B33C MLI: SET NEWLINE <BE70>
B342 WAS "F" OR "R" GIVEN ON COMMAND LINE?
B344 NO >>B34E
B346 YES, POSITION TO SPECIFIED STARTING PT <B57C>
B349 NO ERRORS? >>B34E
B34B IF ERROR, GO CLOSE EXEC >>B2F9
B34E MARK EXEC ACTIVE
B354 AND RETURN TO CALLER
```

```
B355 ********** CLOSE EXEC FILE ***********
```

```
B355 EXEC ACTIVE? (BE43)
B358 NO, SKIP IT >>B365
B35A INDICATE EXEC FILE CLOSING (BE4E)
B35F PICK UP REPNUM FOR EXEC (BC9B)
B362 AND GO CLOSE IT <B4FF>
B365 RETURN
```

```
B366 ********** "VERIFY" COMMAND ***********
```

```
B366 FILE NOT FOUND? >>B3A1
B36B FILE FOUND, WAS A PATHNAME GIVEN?
B36D YES >>B377
B36F NO,
B371 PRINT "(C) APPLE COMPUTER..." <9FC3>
B374 AND A NEW LINE <9FE2>
B377 THEN EXIT
```

---
Beneath Apple ProDOS Supplement

BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84 NEXT OBJECT ADDR: B178

B378 RETURN

B379 ********** FLUSH ALL OPEN FILES ***********************
B379 REFNUM = 0 (ALL FILES)
B378 JUMP INTO FLUSH >>B389

B37D ********** "FLUSH" COMMAND ***********************

B37D ---
B380 WAS PATHNAME1 GIVEN?
B382 NO, FLUSH ALL FILES >>B389
B384 ELSE, LOOK UP NAME IN OPEN FILE LISTS <B479>
B387 NOT AN OPEN FILE >>B391
B389 SAVE REFNUM IN PARM LIST (BDE)
B38E MLI: FLUSH <BE70>
B391 EXIT

B392 ********** "OPEN" COMMAND ***********************

B392 ---
B393 LOOK UP NAME IN OPEN FILE LIST <B479>
B396 NOT CURRENTLY OPEN? >>B3A5
B398 ---
B399 IT IS OPEN, "FILE BUSY" ERROR
B39C RETURN

B39D "FILE TYPE MISMATCH" ERROR
B3A0 RETURN

B3A1 "PATH NOT FOUND" ERROR
B3A3 ---
B3A4 RETURN

B3A5 ---
B3A6_ASSUME "L" IS ZERO
B3A8 😉
B3A9 WAS "L" KEYWORD GIVEN?
B3AA YES, USE HIS VALUE >>B3B7
B3B1 NO, SET "L" TO ZERO (BE68)
B3B8 WAS "T" GIVEN?
B3BB YES, USE HIS TYPE >>B3C5
B3C8 ELSE, DEFAULT TO "TXT"
B3C9 DOES THE FILE ALREADY EXIST? >>B3E8
B3CC NO, "T" GIVEN? IF SO, ERROR >>B3A1
B3CE FORCE TYPE = "TXT" (BEE8)
B3C3 FULL ACCESS (BE7)
B3C4 COPY "L" KEYWORD VALUE (BE5F)
B3D7 TO CREATE (B9A6)
B3D6 AND SET FILE INFO LISTS (BEEA)
B3D3 GO CREATE THE FILE <DAB8>

B445 MAKE INDEX FROM F
B448 GET NAME LENGTH (&)
B44E OR IN DIR FLAG (E)
B451 AND STORE IN OPEN
B457 NAME > OR = TO 30, FIRST <A2S9>
B459 NO... >>B45D ERR CODE
B45B YES, USE 29
B45D STORE THAT AS A LABEL (BEE8)
B462 COPY "L" KEYWORD
B46B ---
B46C COPY FILE NAME TO
B475 COPY ALL OF NAME,

"SINGLY" (BE47)
AS AN INDEX (BDE4)
OF INDEX IN OPEN FILE LIST (BC94)
"CT ADDR: B3E6" (BC5C)
& COUNT AND FALL THRU (B4E0)
"NAME/RECLLEN IN TABLE *****
REFNUM*32 BYTES
B268)
BASIC Interpreter (B1) -- V1.0.1 -- 1 JAN 84 NEXT OBJECT ADDR: B45

ADDR DESCRIPTION/CONTENTS

B455 COMPARE NAMES (B280)
B45B NO MATCH? EXIT WITH 2 FLAG CLEAR >>B4F2
B4BF MATCH, EXIT WITH 2 FLAG SET

B4F3 ********** "CLOSE" COMMAND ****************************

B4F3 ---
B4F6 PATHNAME1 GIVEN?
B4F8 NO, CLOSE ALL FILES >>B54C
B4FA YES, LOOK IT UP IN OPEN FILE TABLES >>B479
B4FD NOT FOUND >>B49B
B4FF FOUND IT, STORE REPNUM IN CLOSE LIST (BDE)
B505 MARK BUFFER PAGE FREE (BC88)
B508 EXEC CLOSING? (BE4E)
B50B YES...NO NEED TO COMpress LISTS >>B529
B50E GET OPEN COUNT (LAST OPENED FILE NO.) (BE4D)
B510 SNAP BUFFERS (BC93)
B51F AND REPNUMS WITH THE LAST OPENED FILE (BC9B)
B529 ---
B52B LEVEL = 0 (BF94)
B530 MLI: CLOSE >>BE70
B533 ERROR? >>B55C
B535 RELEASE THE BUFFER <<A299
B538 EXEC FILE CLOSING? (BE4E)
B53B NO >>B548
B540 YES, EXEC NO LONGER ACTIVE (BE43)
B543 AND NO LONGER CLOSING (BE4E)
B547 RETURN TO CALLER

B548 DROP OPEN FILE COUNT (BE4D)
B54B AND EXIT

B54C ********** CLOSE ALL OPEN FILES *********************

B54C ANY FILES OPEN? (BE4D)
B54F NO >>B55D
B551 YES, EXEC NOT CLOSING (BE4E)
B557 CLOSE LAST FILE OPENED >>B4FF
B55A IF THAT WORKS, START ALL OVER AGAIN >>B54C
B55C EXIT WHEN ALL ARE CLOSED

B55D ---
B55F SET CLOSE REPNUM TO ZERO (ALL FILES) (BDE)
B564 LEVEL = 7 (LEVEL 0 FILES ALREADY CLOSED) (BF94)
B569 EXIT THRU MLI: CLOSE >>BE70

FILENAMES *****************************************************

BASIC Interpreter (B1) -- V1.0.1 -- 1 JAN 84 NEXT OBJECT ADDR: B45

ADDR DESCRIPTION/CONTENTS

B455 COMPARE NAMES (B280)
B45B NO MATCH? EXIT WITH 2 FLAG CLEAR >>B4F2
B4BF MATCH, EXIT WITH 2 FLAG SET

B4F3 ********** "CLOSE" COMMAND ****************************

B4F3 ---
B4F6 PATHNAME1 GIVEN?
B4F8 NO, CLOSE ALL FILES >>B54C
B4FA YES, LOOK IT UP IN OPEN FILE TABLES >>B479
B4FD NOT FOUND >>B49B
B4FF FOUND IT, STORE REPNUM IN CLOSE LIST (BDE)
B505 MARK BUFFER PAGE FREE (BC88)
B508 EXEC CLOSING? (BE4E)
B50B YES...NO NEED TO COMpress LISTS >>B529
B50E GET OPEN COUNT (LAST OPENED FILE NO.) (BE4D)
B510 SNAP BUFFERS (BC93)
B51F AND REPNUMS WITH THE LAST OPENED FILE (BC9B)
B529 ---
B52B LEVEL = 0 (BF94)
B530 MLI: CLOSE >>BE70
B533 ERROR? >>B55C
B535 RELEASE THE BUFFER <<A299
B538 EXEC FILE CLOSING? (BE4E)
B53B NO >>B548
B540 YES, EXEC NO LONGER ACTIVE (BE43)
B543 AND NO LONGER CLOSING (BE4E)
B547 RETURN TO CALLER

B548 DROP OPEN FILE COUNT (BE4D)
B54B AND EXIT

B54C ********** CLOSE ALL OPEN FILES *********************

B54C ANY FILES OPEN? (BE4D)
B54F NO >>B55D
B551 YES, EXEC NOT CLOSING (BE4E)
B557 CLOSE LAST FILE OPENED >>B4FF
B55A IF THAT WORKS, START ALL OVER AGAIN >>B54C
B55C EXIT WHEN ALL ARE CLOSED

B55D ---
B55F SET CLOSE REPNUM TO ZERO (ALL FILES) (BDE)
B564 LEVEL = 7 (LEVEL 0 FILES ALREADY CLOSED) (BF94)
B569 EXIT THRU MLI: CLOSE >>BE70

FILENAMES *****************************************************
BASIC Interpreter

**POSITION** Command

**BEGIN**

B56C   LOC
B56F   NO\$UP NAME OF FILE <B479>
B571   SET USES? >>B509
B574   ANH REM PRINT IN READ/WRIT PARMLIST (B66)
B577   DIG SET NEWLINE LIST (BED2)
B57A   YES FILE? (BE47)
B57C   "$", GET OUT RIGHT NOW! >>B5DA
B57D   "E", GET OUT RIGHT NOW! >>B5DA
B581   NO OR "R" GIVEN? (BE57)
B583   BOH INVALID PARM >>B5D7
B585   YES GIVEN?
B587   JUG INVALID PARM >>B5D7
B589   NO OR "R" GIVEN?
B58B   JUG "R" GIVEN?
B58E   (ST "ST", COPY "F" VALUE TO "F" (BE65)
B597   SET "E" AND "F" ARE ALIASES) (BE63)
B5A5   BURN COUNT TO 239 (MAXIMUM LINE LEN)
B5A9   --FIL IS AT $280 (BED8)
B5AE   NO:
B5B0   MLI LIST CHAR IS EITHER $0D OR $0D (BED3)
B5B3   ERR "B", "E", OR "R" GIVEN?
B5BD   >>B59

**SKIP LINES BY READING THEM *****

B5B5   --
B5B8   "F"
B5BC   YES? >>B64
B5BE   EOL, DONE >>B5DA
B5C0   MLK...
B5C3   ERR "READ NEXT FIELD (LINE) <BE70>
B5CE   ERR "EOL" >>B59
B5D5   ANCR PRT R VALUE BY ONE
B5D7   "I"
B5DA   "E", "F", "R" ERROR POSITIONING? >>B685
B5DB   "ASSUME "$L" = 239.
B5DF   "$L" GIVEN?
B5E0   NO >>B666
B5E3   YES, USE HIS "$L" VALUE (BE5F)
B5E5   UNLESS ITS $256 >>B5BB
B5F0   OR $256 >>B5BB
B5F4   DOUBLE QUOTE IT SO COMMAS COME THRU (BED7)
B5F6   READ INTO $281
B5F8   IF NO "$L", READ TO $280 (BED7)
B5FB   ML CHAR = $0D/$0D (OR NONE IF "$L") (BED7)
B5FC   MLI SET NEWLINE <BE70>
B5FD   ERROR? >>B685
B5FE   MARK INPUT "READ" FILE ACTVE (BE44)
B600   AND RETURN

**READ DIR FILE *******

B605   SET READ/WRITE LIST REMPRINT (B66)
B609   AND GET/SET LIST REMPRINT (B6C)
B60E   READING TO $259 (BED7)
B610   INIT CAT FLAG TO FIRST LINE VALUE (BE6F)
B615   YES? >>GIVEN?
B616   NO, DONE >>B5BB
B61E   YES, ZERO OUT MARK (B6C8)
B622   MLK "REWIND FILE <BE70>
BASIC Interpreter (B1) -- V1.0.1 -- 1 JAN 84 NEXT OBJECT ADDR: B6B3

ADDR DESCRIPTION/CONTENTS
B6B3 ERROR? >>B6BA
B6B7 MARK INPUT FILE ACTIVE (BE44)
B6BA AND EXIT

B6BB ************ "RANGE ERROR" ****************************

B6BB "RANGE ERROR" CODE
B6BF EXIT TO CALLER

B6C0 ************ PRE-POSITION FOR I/O ***********************

B6C0 --
B6C3 "b", "f", or "r" GIVEN?
B6C5 NO, EXIT >>B709
B6C7 "r"?
B6C9 NO >>B6D5
B6CB YES, COMPUTE ABSOLUTE POSITION <B5DB>
B6CE ERROR? >>B6BB
B6D0 NO, SET MARK TO NEW POSITION <B702>
B6D3 ERROR? >>B70A
B6D5 "f" GIVEN? (BE57)
B6DA NO >>B6E1
B6DC SKIP LINES UNTIL "f" = 0 <B5A9>
B6DF ERROR? >>B70A
B6E1 "b" GIVEN? (BE57)
B6E6 NO >>B709
B6EA MLI: GET MARK <BE70>
B6ED ERROR? >>B70A
B6F3 ADD "b" VALUE TO CURRENT MARK (BE5A)
B6F6 (3 BYTE ADD) (BE80)
B700 OVERFLOW? >>B6BB
B702 --
B704 MLI: SET MARK <BE70>
B707 ERROR? >>B70A
B709 --
B70A --
B70C EXIT TO CALLER

B70D ************ "WRITE" COMMAND ****************************

B70D LOOKUP OPEN FILE NAME <B479>
B710 NOT AN OPEN FILE? >>B722
B712 STORE READ/WHITE RENNUM (BED6)
B715 AND GET/SET RENNUM (BEC7)
B718 AND NEWLINE RENNUM IN PARM LISTS (BED2)
B71A DIR FILE? (BE47)
B71B NO, OK >>B724

B72A YES, "FILE" >>B722
B72B --
B72D EXIT TO CALL

B72F "CHECKED" ERROR
B730 YES, "FILE" >>B722
B732 --
B733 EXIT TO CALL

B734 DATA BUFFER FOR "b", "f", AND "r" <B6C0>
B736 PRE-POSITION <B747>
B738 NO ERRORS? "RANGE ERROR?"
B73A WAS ERROR ADJ >>B722
B73C NO, RETURN ERROR OR MLI'S?
B73E YES, "r" VALUE?
B739 MLI'S...SET <B707>
B73D MLI: SET EOF?
B73F ERROR? >>B722 AGAIN TO SET MARK <B6D0>
B740 AND THEN "R" GIVE UP >>B722
B745 ERROR? THEN HIMEM
B747 BUFFER IS AT "WRITE" FILE ACTIVE (BE45)
B753 INICATES MLINER
B757 RETURN TO C

B758 *********** "API*

B758 --
B75A LOOK UP NAME <B76A>
B75C FOUND? >>FIRST <B392>
B75E NO, OPEN IT? B
B760 GET <B707-
B764 NO, RENUM <B76B
B767 YES, OK >>B7E11
B769 ELSE, BREAK.
B76A --
B76D WRITE PARM LIST (BED6)
B76F RENNUM TO PARM LIST (BEC7)
B770 AND GET/SET (847)
B771 DIR FILE? (B
B774 NO >>B77A

B776 YES, "FILE" >>B722
B778 --
B779 EXIT TO CALL

B77A PICK UP "l", "m", "n", "r", "w" VALUE (BE5F)
B77A PICK UP "t", "u", "v", "w" VALUE (BE5F)
B781 DID USER END?
B785 YES... >>B787 STS CURRENT "l" VALUE (BCA4)
B787 NO, "FILE"
B78D --
B78D "m*"32 FOR INDEX INTO
B792 COMPUTE RENNUM
B793 FILE NAME TO "l" VALUE IN OPEN FILE (BCF7)
B794 SAVE CURRENT "d" IN CURRENT RELEN (BCA4)
B79B NAME TAG'S <B707>
B797 MLI: GET EOF
BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84 NEXT OBJECT ADDR: B7AA

ADDRESSES CONTENTS

79A    ERROR? >>B778

79B    NO >>B768

79C    YES >>B7BD

------

B768    NO, FORCE TO RECORD BOUNDARY <B7C0>

B76B    ERROR? >>B778

B76D    ELSE, GO SET EOF=MARK/OUTPUT FILE ACTIVE >>B73B

B76F    ********** FORCE TO EVEN RECORD BOUNDARY **************

B76F    (FIND RECORD NUMBER OF THIS POSITION)

B76F

B772    ---

B773    COPY EOF TO ACCUM (BEC7)

B775    CLEAR MSB'S (BCB2)

B778    GET READY FOR A 24 BIT DIVIDE

B77B    DIVIDE EOF BY... <AB17>

B77C    RECORD LENGTH (BCA4)

B77F    ---

B780    WAS THERE A REMAINDER? (BCB3)

B782    YES, CURRENT RECORD LEN LESS REMAINDER (BCB2)

B784    PLUS OLD EOF MARK (BCB8)

B787    GIVES NEW EOF ON AN EVEN RECORD BOUNDARY (BEC9)

B789    "RANGE ERROR" POSSIBLE IF OVERFLOW OCCURS

B792    RETURN TO CALLER

B794

B795    ********** GET FILE INFO **************

B796    SET NUMBER OF PARMS (18)

B79F    MLI CODE FOR GET FILE INFO

B802    DO IT >>B848

B803    ********** SET FILE INFO ***************

B804    MODIFIED TIME/DATE = 0

B807    SET NUMBER OF PARMS (7)

B80A    MLI CODE FOR SET FILE INFO

B80D    EXIT THRU MLI: GET/SET FILE INFO >>B700

B80E    ********** BI I/O INDIRECTION VECTORS **************

B812    DOSOUT VECTOR >>BE38

B814    DOSIN VECTOR >>BE3A

B816    ********** STATE I/O VECTORS TABLE **************

B818    IMMEDIATE MODE (STATE=0) CSWL/KSWL

B81B    DEFERRED MODE (STATE=4) CSWL/KSWL

B81E    (STATE=8) CSWL/KSWL

B820    (STATE=C) CSWL

B823

B825

B85D    ********** SYSTBL ***********************

B85D    LSB'S OF MLI CALL PARAMETER LISTS IN THE

B85D    BI GLOBAL PAGE ($BEXX)

B860    CREATE: SAW DESTROY: SAC RENAME: SAF

B862    SFI: SB4 GFI: SB4 ONLINE: SC6

B865    SPFX: SAC GPFX: SAC OPEN: SCB

B868    NEWLINE:SD1 READ: SD5 WRITE: SD5

B86B    CLOSE: SDD FLUSH: SDD SMARK: SC6

B86E    GMARK: SC6 SEOF: SC6 GEOF: SC6

B871    SBUF: SC6 GBUF: SC6

B873    ********** APPELSOFT TOKENS ******************

B873    TOKENS REQUIRING SPECIAL ATTENTION HAVE THEIR MSB OFF AND ARE AN OFFSET FROM A

B873    JMP IN THE TRACE HANDLER IN THE BI

B876    FIRST IS SB0 (END)

B879    CALL

B881    TRACÉ, NOTRACE, NORMAL

B883    INVERSE, FLASH

B885    RESUME

B887    LET, IF

B889    PRINT, LIST

B893    ********** COMMAND NAME TABLES **************

B896    OFFSETS TO LAST CHARACTER OF EACH COMMAND

B898    NAME IN THE COMMAND NAME TABLE BELOW.

B89B    COMMANDS ARE ARRANGED ACCORDING TO LENGTH

B89E    WITH THREE BYTES NAMES FIRST. IF THE MSB

B8A1    OF AN INDEX IS ON, THEN THIS IS THE LAST

B8A4    NAME OF THE GIVEN LENGTH (NEXT WILL BE

B8A6    ONE BYTE LONGER).

B8A9

B8B3    01 IN# 02 PR# 03 CAT

B8BA    04 FRE 05 RUN 06 BRU

B8BB    07 EXEC 08 LOAD 09 LOCK

B8BC    0A OPEN 0B READ 0C SAVE

B8BF    0D BLOAD 0E BSAVE 0F CHAN

B8C2    10 CLOSE 11 FLUSH 12 MONON

B8C5    13 STORE 14 WRITE 15 APPEND

B8C8    16 CREATE 17 DELETE 18 PREFIX

B8CB    19 RENAME 1A UNLOCK 1B VERIFY

B8CE    1C CATALOG 1D RESTORE 1E POSITION

B8D1    'BSAVERIFY'LOADDELETECATALOGOPENWR

B8F1    'ITEXECREATEPRESTORENAMEMBRUNLOCKC'

B911    'MAIN#FLUSHREADPOSITIONMOPR#PRE'

B931    'FIXCLOSEAPPEND'
B93F ********** COMMAND HANDLER ADDRESS TABLE **************

ADDRESSES OF THE COMMAND HANDLER ROUTINES
FOR EACH COMMAND IN THE ORDER GIVEN ABOVE.

B941 IN#
B943 FR#
B945 CAT
B947 FRE
B949 RUN
B94B BRUN
B94D EXEC
B94F LOAD
B951 LOCK
B953 OPEN
B955 READ
B957 SAVE
B959 BLOAD
B95B BSAVE
B95D CHAIN
B95F CLOSE
B961 FLUSH
B963 NOMON
B965 STORE
B967 WRITE
B969 APPEND
B96B CREATE
B96D DELETE
B96F PREFIX
B971 RENAME
B973 UNLOCK
B975 VERIFY
B977 CATALOG
B979 RESTORE
B97B POSITION
B97D "-" COMMAND

B97F ********** 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.
0000 = FETCH PREFIX, PATHNAME OPTIONAL
0001 = SLOT (FOR PR# OR IN#)
0002 = DEFERRED COMMAND ONLY
1000 = PATHNAME IS OPTIONAL
0000 = IF FILE NOT FOUND, CREATE IT
0040 = "T" (FILE TYPE) PERMITTED
0020 = PATHNAME2 (RENAME) PERMITTED
0100 = PATHNAME1 EXPECTED
0000 = "A" (ADDRESS) PERMITTED

0040 = "R" (BYTE) PERMITTED
0020 = "R" (END ADDRESS) PERMITTED
0010 = "L" (LENGTH) PERMITTED
0008 = "S" (LINE NO.) PERMITTED
0004 = "S" AND/OR "D" (SLOT/DRIVE)
0002 = "P" (FIELD) PERMITTED
0001 = "R" (RECORD) PERMITTED
("V" IS IGNORED)
BASIC Interpreter (B1) -- V1.0.1 -- 1 JAN 84  NEXT OBJECT ADDR: B9E8

---

B98D **************************** KEYWORD NAME TABLE ****************************

B98D  'ABELSEDFRVF'

B9C7 **************************** KEYWORD BIT POSITION TABLE ******************************

BIT POSITIONS IN PERMITTED PARMs TABLE
FOR EACH KEYWORD IN THE ORDER GIVEN IN
NAME TABLE. "V" IS 00 (NOT USED)

B9C7 ---

B9D1 **************************** KEYWORD SIZE/OFFSET TABLE ****************************

LOW 2 BITS - SIZE-1 OF VALUE IN BYTES
HIGH 6 BITS - OFFSET TO LAST BYTE OF VALUE
FROM $8E58

B9D1 A: 2 BYTES AT +1
B9D2 B: 3 BYTES AT +4
B9D3 C: 2 BYTES AT +6
B9D4 D: 2 BYTES AT +8
B9D5 E: 1 BYTE AT +9
B9D6 D: 1 BYTE AT +A
B9D7 F: 2 BYTES AT +C
B9D8 B: 2 BYTES AT +E
B9D9 V: 1 BYTE AT +10 (IGNORED)
B9DA @: 2 BYTES AT +11

B9DB **************************** FILE TYPES TABLES ****************************

FILE TYPE CODES, GIVEN IN INVERSE ORDER
TO FILE TYPE NAMES WHICH FOLLOW.

B9DB $8F = "SYS"
B9DC $8E = "REL"
B9DD $8D = "VAR"
B9DE $8C = "BAS"
B9DF $8B = "IVR"
B9E0 $8A = "INT"
B9E1 $89 = "CMD"
B9E2 $88 = "DIR"
B9E3 $86 = "BIN"
B9E4 $84 = "TXT"
B9E5 $83 = "PAS"
B9E6 $1A = "AWP"
B9E7 $1B = "ASP"
B9E8 $19 = "ADB"
BASIC Interpreter (BI) -- V1.0.1 -- 1 JAN 84  NEXT OBJECT ADDR: BACE
BADA "DESCRIPTION/CONTENTS"
BAE5 ""
BAEE ""
BAFE "LOCKS FREE:;TAB$(16)
BAF5 "LOCKS USED:;TAB$(2C)
BAF0 "L BLOCKS:"
BB00 "MARK"
BB09 "HUGE ERROR"
BB0F "PAC"
BB12 "NOTE PROTECTED"
BB18 "I/O OF DATA"
BB1E "NOT FOUND"
BB24 "FILE ERROR"
BB2B "INK"
BB35 "RAM FULL"
BB42 "FILE""LID PARAMETER"
BB4E "FILE TOO LARGE"
BB59 "NO"""FILE TYPE MISMATCH"
BB63 "SYM""GRAM TOO LARGE"
BB6B "DISK"
BB73 "FILE"
BB7B "DEAX ERROR"
BB86 "FACTORY FULL"
BB8D "FILE NOT OPEN"
BB99 "FILE""ICATE FILE NAME"
BB99 "FILE BUSY"
BB9A "FILE(S) STILL OPEN"
BB9E "BELL"
BB9F "PAUSE MESSAGE"
Beneath Apple ProDOS Supplement

BASIC Interpreter (B1) -- V1.0.1 -- 1 JAN 84   NEXT OBJECT

ADDRESS DESCRIPTION/CONTENTS

BCB3  MONTH
BCB4  DAY
BCB5  YEAR
BCB6  ERROR MSG LEN OR LINE LEN FOR CAT/CATALOG
BCB7  ENTRY LENGTH IN DIRECTORY FILE
BCB8  ENTRIES PER BLOCK IN DIRECTORY FILE
BCB9  FILE COUNT FROM DIRECTORY FILE
BCBB  DIRECTORY ENTRY NUMBER COUNTER

BCBC  ********** PATHNAME 1 BUFFER *********************

BCBC  COMMAND OR PATH LENGTH
BCBD  TXBUF (COMMAND OR PATHNAME STRING)
BCF0  NOT USED

BCFE  ********** OPEN FILE NAME TABLE **************
(EACH ENTRY IS 32 BYTES LONG)
(THERE ARE 8 ENTRIES)

BCFE  FILE 0: LENGTH OF NAME
BCFP  FILE 0: L VALUE LSB
BD00  FILE 0: L VALUE MSB
BD01  FILE 0: START OF NAME STRING
(FILE NAME IS STORED BACKWARDS)

******

******
### 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-BE05</td>
<td>BI.ENTRY</td>
<td>JMP to WARMDOSS (BI warmstart vector).</td>
</tr>
<tr>
<td>BE06-BE0B</td>
<td>DOSCMD</td>
<td>JMP to SYNTAX (BI command line parse and execute).</td>
</tr>
<tr>
<td>BE0C-BE0E</td>
<td>ERETT</td>
<td>JMP to user-installed external command parser.</td>
</tr>
<tr>
<td>BE0F</td>
<td>ERRCODE</td>
<td>JMP to BI error handler.</td>
</tr>
<tr>
<td>BE10-BE1F</td>
<td>OUTVEC</td>
<td>Place error number in A-register.</td>
</tr>
<tr>
<td>BE12-BE1F</td>
<td>INVEC</td>
<td>ProDOS error code (also at $DE, Applesoft ONERR code).</td>
</tr>
<tr>
<td>BE30-BE31</td>
<td>VECTOUT</td>
<td>Default output vector in monitor and for each slot (1-7).</td>
</tr>
<tr>
<td>BE32-BE33</td>
<td>VECTIN</td>
<td>Default input vector.</td>
</tr>
<tr>
<td>BE34-BE35</td>
<td>VDOSIO</td>
<td>BI's output intercept address.</td>
</tr>
<tr>
<td>BE36-BE37</td>
<td>VDOSIO</td>
<td>BI's input intercept address.</td>
</tr>
<tr>
<td>BE38-BE3B</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>BE4F</td>
<td>PREGX</td>
<td>X-register savearea.</td>
</tr>
<tr>
<td>BE46</td>
<td>DTRACE</td>
<td>Y-register savearea.</td>
</tr>
<tr>
<td>BE48</td>
<td>DTSTATE</td>
<td>Applesoft TRACE is enabled flag (MSB on).</td>
</tr>
<tr>
<td>BE4C</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>IFILACTV</td>
<td>READ file active flag (MSB on).</td>
</tr>
<tr>
<td>BE45</td>
<td>OFLACTV</td>
<td>WRITE file active flag (MSB on).</td>
</tr>
<tr>
<td>BE46</td>
<td>PXFILACTV</td>
<td>PREFIX read active flag (MSB on).</td>
</tr>
<tr>
<td>BE47</td>
<td>DIRFLG</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>TBUPFTR</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>OPENCNT</td>
<td>Number of files open (not counting EXEC).</td>
</tr>
<tr>
<td>BE4E</td>
<td>YYFILE</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-BE51</td>
<td>XTRNADDR</td>
<td>External command handler address.</td>
</tr>
<tr>
<td>BE52</td>
<td>XLEN</td>
<td>Length of command name (less one).</td>
</tr>
<tr>
<td>ADDR</td>
<td>LABEL</td>
<td>CONTENTS</td>
</tr>
<tr>
<td>-------</td>
<td>-------</td>
<td>----------</td>
</tr>
<tr>
<td>BE53</td>
<td>LABEL</td>
<td>CONTENTS</td>
</tr>
<tr>
<td>00</td>
<td>Number of command:</td>
<td></td>
</tr>
<tr>
<td>S00</td>
<td>external</td>
<td>S0A = OPEN</td>
</tr>
<tr>
<td>S01</td>
<td>IN$</td>
<td>S0B = READ</td>
</tr>
<tr>
<td>S02</td>
<td>PR#</td>
<td>S0C = SAVE</td>
</tr>
<tr>
<td>S03</td>
<td>CAT</td>
<td>S0D = BLOAD</td>
</tr>
<tr>
<td>S04</td>
<td>FREE</td>
<td>S0E = SAVE</td>
</tr>
<tr>
<td>S05</td>
<td>RUN</td>
<td>$19 = CHAIN</td>
</tr>
<tr>
<td>S06</td>
<td>BRUN</td>
<td>$10 = CLOSE</td>
</tr>
<tr>
<td>S07</td>
<td>EXEC</td>
<td>$11 = FLUSH</td>
</tr>
<tr>
<td>S08</td>
<td>LOAD</td>
<td>$12 = NOMON</td>
</tr>
<tr>
<td>S09</td>
<td>SAVE</td>
<td>$13 = STORE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$1E = POSITION</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ADDR</th>
<th>LABEL</th>
<th>CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>BE55</td>
<td>PBITS</td>
<td>Permitted command operands bits:</td>
</tr>
<tr>
<td>S8000</td>
<td>Prefix needed. Pathname optional.</td>
<td></td>
</tr>
<tr>
<td>S4000</td>
<td>$lot number only (PR# or IN$).</td>
<td></td>
</tr>
<tr>
<td>S2000</td>
<td>Deferred command.</td>
<td></td>
</tr>
<tr>
<td>S1000</td>
<td>File name optional.</td>
<td></td>
</tr>
<tr>
<td>S0100</td>
<td>If file does not exist, create it.</td>
<td></td>
</tr>
<tr>
<td>S0000</td>
<td>T: file type permitted.</td>
<td></td>
</tr>
<tr>
<td>S0000</td>
<td>$2: second file name required.</td>
<td></td>
</tr>
<tr>
<td>S0000</td>
<td>First file name required.</td>
<td></td>
</tr>
<tr>
<td>S0000</td>
<td>AD: address keyword permitted.</td>
<td></td>
</tr>
<tr>
<td>S0000</td>
<td>B: byte offset permitted.</td>
<td></td>
</tr>
<tr>
<td>S0000</td>
<td>E: ending address permitted.</td>
<td></td>
</tr>
<tr>
<td>S0010</td>
<td>L: length permitted.</td>
<td></td>
</tr>
<tr>
<td>S0000</td>
<td>$2: line number permitted.</td>
<td></td>
</tr>
<tr>
<td>S0004</td>
<td>S or D: slot/drive permitted.</td>
<td></td>
</tr>
<tr>
<td>S0002</td>
<td>F: field permitted.</td>
<td></td>
</tr>
<tr>
<td>S0001</td>
<td>R: record permitted.</td>
<td></td>
</tr>
<tr>
<td>(Always permitted but ignored.)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ADDR</th>
<th>LABEL</th>
<th>CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>BE57</td>
<td>FBITS</td>
<td>Operands found on command line. Same bit assignments as above.</td>
</tr>
<tr>
<td>BE58</td>
<td>A</td>
<td>VADDR</td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>VBYTE</td>
</tr>
<tr>
<td>BE60</td>
<td>A</td>
<td>VENDA</td>
</tr>
<tr>
<td>BE62</td>
<td>E</td>
<td>VLINT</td>
</tr>
<tr>
<td>BE63</td>
<td>S</td>
<td>VSLOT</td>
</tr>
<tr>
<td>BE64</td>
<td>D</td>
<td>VDRIK</td>
</tr>
<tr>
<td>BE66</td>
<td>F</td>
<td>VFELD</td>
</tr>
<tr>
<td>BE67</td>
<td>R</td>
<td>VRKCD</td>
</tr>
<tr>
<td>BE69</td>
<td>V</td>
<td>WVOLM</td>
</tr>
<tr>
<td>BE6A</td>
<td>T</td>
<td>VLINE</td>
</tr>
<tr>
<td>BE6C</td>
<td>PR#</td>
<td>VTYPE</td>
</tr>
<tr>
<td>BE6D</td>
<td>IN$</td>
<td>VIOSLT</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ADDR</th>
<th>LABEL</th>
<th>CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>BE66</td>
<td>VPATH1</td>
<td>Primary pathname buffer (address of length byte).</td>
</tr>
<tr>
<td>BE6F</td>
<td>VPATH2</td>
<td>Secondary pathname buffer (address of length byte).</td>
</tr>
<tr>
<td>BE70</td>
<td>GOSYSTEM</td>
<td>Call the MI using the parameter tables which follow.</td>
</tr>
<tr>
<td>BE85</td>
<td>SYSCALL</td>
<td>MLI call number for this call.</td>
</tr>
<tr>
<td>BE86</td>
<td>SYSPARM</td>
<td>Address of MLI parameter list for this call.</td>
</tr>
<tr>
<td>BE88</td>
<td>BP6A</td>
<td>Return from MLI call.</td>
</tr>
<tr>
<td>BE88-8BE9</td>
<td>BP6C</td>
<td>MLI error return: translate error code to B1 error number.</td>
</tr>
<tr>
<td>BE9F</td>
<td>BISPAREL</td>
<td>Not used.</td>
</tr>
<tr>
<td>BEA0</td>
<td>SCREATE</td>
<td>CREATE parameter list.</td>
</tr>
<tr>
<td>BEAC</td>
<td>SGPRFX</td>
<td>SET_PREFIX, SET_PREFIX, DESTROY parameter list.</td>
</tr>
<tr>
<td>BB84</td>
<td>BB5C</td>
<td>SRENAMENAMES parameter list.</td>
</tr>
<tr>
<td>BB84-8BE5</td>
<td>BB5C</td>
<td>SET_FILE_INFO, SET_FILE_INFO parameter list.</td>
</tr>
<tr>
<td>BEC6</td>
<td>BECA</td>
<td>ONLINE, SET_MARK, GET_MARK, SET_EOF, GET_EOF, SET_BUF, GET_BUF, QUIT parameter list.</td>
</tr>
<tr>
<td>BECC</td>
<td>BB5D</td>
<td>OPEN parameter list.</td>
</tr>
<tr>
<td>BED1</td>
<td>BB5H</td>
<td>SET_NEWLINE parameter list.</td>
</tr>
<tr>
<td>BED5</td>
<td>BB5C</td>
<td>READ, WRITE parameter list.</td>
</tr>
<tr>
<td>BB6D</td>
<td>BB5C</td>
<td>CLOSE, FLUSH parameter list.</td>
</tr>
<tr>
<td>BB6F</td>
<td>BB5C</td>
<td>&quot;COPYRIGHT APPLE, 1983&quot;</td>
</tr>
<tr>
<td>BB5F</td>
<td>BB5C</td>
<td>GETBUF hello buffer allocation subroutine vector.</td>
</tr>
<tr>
<td>BEF8</td>
<td>BB5C</td>
<td>FREEBUF buffer free subroutine vector.</td>
</tr>
<tr>
<td>BEFB</td>
<td>BB5C</td>
<td>Original HIMEM MSB. Not used.</td>
</tr>
<tr>
<td>BEFC</td>
<td>BB5C</td>
<td></td>
</tr>
</tbody>
</table>
ProDOS VERSION 1.0.2

In March, 1984, Apple began shipping Version 1.0.2 of ProDOS along with the Apple IIc. Version 1.0.2 is also the base for some of Apple's own software, such as AppleWorks. The differences between this version and its predecessor, Version 1.0.1, are minor. Except for the specific areas mentioned below, the description of Version 1.0.1 in this Supplement may be used for Version 1.0.2.

ProDOS Loader

Version 1.0.2 is identical to Version 1.0.1.

ProDOS Relocator

Replace the comments at the following addresses:

20A2: YES, QUIT VECTOR -->$EEDB
21B8: LEN = $1ED7
21C1: TO = $AF71
21C5: FRM = $AF71
249A: 'PRODOS 1.0.2 15-FEB-84'

All other addresses and comments remain the same as Version 1.0.1.

ProDOS MLI (Kernel)

Replace the comments at the following addresses:

D19A: Indicate error type 2
DE6D: Stomp on $F300+$5E

At $E948, 12 bytes are added. This causes all addresses greater than $E947 and all references to those addresses to be increased by $0C. For example, all references to $E948 in Version 1.0.1 become $E954 in Version 1.0.2. The 12-byte insertion is commented as follows:

E948: Flush file; update directory <E71C>
E94B: No error? >>E954
E94D: Error, return error code

ProDOS System Global Page

Version 1.0.2 is identical to Version 1.0.1.
ProDOS Quit Code

Version 1.0.2 is identical to Version 1.0.1. There is different data due to different uninitialized variables in a data area at the end of the Quit Code section, but this has no effect on the operation of the software.

ProDOS Disk II Device Driver

Minor changes to the beginning of the Disk II Device Driver caused the area from $F800 to $F8F3 to change and added a routine at the end of the Version 1.0.1 code ($FEBE to $FED1). These two areas are described on the following pages. The rest of the Disk II Device Driver is identical to Version 1.0.1.

ProDOS IRQ Handler

Version 1.0.2 is identical to Version 1.0.1.

ProDOS BI Relocator

Version 1.0.2 is identical to Version 1.0.1.

ProDOS BASIC Interpreter (BI)

Version 1.0.2 is identical to Version 1.0.1.

ProDOS BI Global Page

Version 1.0.2 is identical to Version 1.0.1.
### Disk II Device Driver -- V1.0.2 -- 15 FEB 84  
**NEXT OBJECT ADDR: F89F**

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>F8E2</td>
<td>Read the data - Good read? (FBFD)</td>
</tr>
<tr>
<td>F8E5</td>
<td>No, then try again &gt;&gt; F8A4</td>
</tr>
<tr>
<td>F8E7</td>
<td>Indicate no errors</td>
</tr>
<tr>
<td>F8E9</td>
<td>BNE Instruction, never taken</td>
</tr>
<tr>
<td>F8EA</td>
<td>Indicate error</td>
</tr>
<tr>
<td>F8EB</td>
<td>Preserve error number (FB58)</td>
</tr>
<tr>
<td>F8EE</td>
<td>Get Slot</td>
</tr>
<tr>
<td>F8F0</td>
<td>Turn motor off (C088)</td>
</tr>
<tr>
<td>F8F3</td>
<td>Return to caller</td>
</tr>
</tbody>
</table>

### Disk II Device Driver -- V1.0.2 -- 15 FEB 84  
**NEXT OBJECT ADDR: FEBB**

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

| FEBE  | Get Block Number                      |
| FEC2  | Is Block Number good? (FB56)          |
| FEC5  | Yes, if less than $100 >> FED0        |
| FEC8  | No, if greater than or equal to $200 >> FECE |
| FECC  | No, if greater than or equal to $118 >> FED0 |
| FECE  | Indicate error                        |
| FECF  | Return to caller                      |
| FED0  | All is well                           |
| FED1  | Return to caller                      |
| FED2  | Unused up to $FF00 >> 02E             |
ERRATA TO BEneath Apple ProDOS

Please make the following corrections to your copy of Beneath Apple ProDOS:

Page 3-10:

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 "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
...
WE'VE GOT IT NOW

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
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 7-26:

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
```

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...
1...
1...
1...
1...

Reference Card, Panel 9

The last entry for "MLI ERROR CODES" should be:

$5A Bad vol. bit map

(not $58).
ORDERING FUTURE SUPPLEMENTS

New supplements will be published to reflect the changes made as ProDOS is updated. To order an updated supplement, mail the coupon on the next page directly to Quality Software (at the address on the coupon), along with a payment of $10.00 plus shipping and handling charges.* Your payment can be a check or bank draft in US dollars, or your VISA or MASTERCARD number and expiration date. California residents must add the appropriate sales tax (6 or 6.5%). No phone orders or CODs will be accepted.

footnote:

SHIPPING CHARGES

*SHIPPING

United States...........................$ 2.50
All other countries (insured air mail)..................$10.00