SUPPLEMENT TO

Beneath Apple ProDOS

For ProDOS Version 1.1.1

by Don D. Worth and Pieter M. Lechner

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

Beneath Apple ProDOS
by Don Worth & Pieter Lechner

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

Beneath Apple DOS
by Don Worth & Pieter Lechner

Understanding the Apple II
by Jim Sather

Understanding the Apple IIe
by Jim Sather

$19.95

$10.00

$19.95

$22.95

$24.95

Apple Utility Software from Quality Software

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

Universal File Conversion (includes diskette)
by Gary Charpentier

$49.95

$34.95

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

Illustrations by George Garcia

(c)1986 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><strong>PRODOS, VERSION 1.1.1</strong></td>
<td></td>
</tr>
<tr>
<td>How ProDOS 1.1.1 is Loaded and Relocated</td>
<td>6</td>
</tr>
<tr>
<td>ProDOS Loader</td>
<td>7</td>
</tr>
<tr>
<td>ProDOS Relocator</td>
<td>10</td>
</tr>
<tr>
<td>Relocation routines</td>
<td></td>
</tr>
<tr>
<td>RAMdrive Device Driver</td>
<td></td>
</tr>
<tr>
<td>SYSTEM File loader</td>
<td></td>
</tr>
<tr>
<td>ProDOS MLI (Kernel)</td>
<td>23</td>
</tr>
<tr>
<td>ProDOS System Global Page</td>
<td>57</td>
</tr>
<tr>
<td>ProDOS Quit Code</td>
<td>59</td>
</tr>
<tr>
<td>ProDOS Disk II Device Driver</td>
<td>63</td>
</tr>
<tr>
<td>ProDOS IRQ Handler</td>
<td>70</td>
</tr>
<tr>
<td><strong>BASIC.SYSTEM, VERSION 1.1</strong></td>
<td></td>
</tr>
<tr>
<td>How BASIC.SYSTEM is Loaded and Relocated</td>
<td>71</td>
</tr>
<tr>
<td>BI Relocator</td>
<td>72</td>
</tr>
<tr>
<td>BASIC Interpreter (BI)</td>
<td>75</td>
</tr>
<tr>
<td>BI Global Page</td>
<td>110</td>
</tr>
<tr>
<td><strong>DISK II CONTROLLER BOOT ROM</strong></td>
<td></td>
</tr>
<tr>
<td>Disk II Controller Card--Apple II/II+/IIe</td>
<td>112</td>
</tr>
<tr>
<td>Disk II Controller--Apple IIc</td>
<td>114</td>
</tr>
<tr>
<td><strong>ERRATA</strong></td>
<td></td>
</tr>
<tr>
<td>Errata to Beneath Apple ProDOS</td>
<td></td>
</tr>
<tr>
<td>1st printing, 1984</td>
<td>117</td>
</tr>
<tr>
<td>2nd printing, 1985</td>
<td>122</td>
</tr>
</tbody>
</table>
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. All assembly language programmers will find this supplement useful in learning about how an operating system works. This information is presented in the spirit of helping the user to better understand how ProDOS works. The authors do not endorse indiscriminant modification of the ProDOS components. Whenever possible, standardized interfaces to ProDOS should be used to avoid the uncontrolled modifications which plagued Apple's previous operating system, DOS 3.3.

External system programs and utilities such as the FILER and CONVERT are not covered here, nor are disk controller ROM's other than the 5.25" controllers sold by Apple.

The information presented here is for the release of the ProDOS operating system called Version 1.1.1. A previous supplement to Beneath Apple ProDOS documented the structure of Versions 1.0.1 and 1.0.2 of ProDOS.

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.
HOW "PRODOS" (Version 1.1.1) IS LOADED AND RELOCATED

1. The ProDOS Loader (or a "=" command) loads the "PRODOS" file to memory address $2000, and jumps to it.

   I----------- I
   I "PRODOS" I
   I 30 BLOCK FILE I
   I (29 data blocks I plus one index I--->
   I L$3A00 I
   I I
   I I
   I I
   I I
   I I
   I I
   I I
   I I
   I

2. Copy from within Relocator to low memory:
   SYSTEM file Loader
   PAGE 3 IMAGE
   80-COL CARD CHECKER

3. Copy to HIRAM:
   IRQ Handler
   System Global Page
   MLI Kernel
   Disk II Device Driver

4. Final moves:
   FUNCTION FROM TO LENGTH
   Clock code 4F00 D742 7D
   QUIT code 5700 D100* 300
   RAM drive...
   Caller 2C00 FF00 9A
   Driver 2A00 200** 200

*BANK2 **AUX MEMORY
**ProDOS Loader -- V1.1.1 -- 18 SEP 84**

---

**ADDRESSES/DESCRIPTION/CONTENTS**

---

0800 **MODULE STARTING ADDRESS**

- **PRODOS LOADER**
- **THIS CODE IS LOADED FROM BLOCK 0.**
- **INTO MEMORY AT $800.**
- **ITS PURPOSE IS TO LOAD THE "PRODOS"**
- **FILE INTO $2000 AND JUMP TO IT.**
- **(PRODOS RELOCATOR IS AT $2400)**
- **VERSION 1.1.1 -- 18 SEP 84**
- **(THE LOADER IS STILL THE SAME AS IT**
- **WAS IN VERSION 1.0.1)**

---

***EXTERNAL ADDRESSES***

0027 **ROM BOOT SUBRNM BUFFER PAGE ADDR**

0028 **ROM BOOT SUBRNM SLOT * 16**

003D **ROM BOOT SUBRNM SECTOR TO READ**

0040 **ROM BOOT SUBRNM CURRENT TRACK**

0041 **ROM BOOT SUBRNM TRACK TO READ**

- **BLOCK READ PARAMETER LIST**

0042 **COMMAND NUMBER (1 = READ)**

0043 **SLOT * 16**

0044 **I/O BUFFER ADDRESS ($44/$45)**

0046 **BLOCK TO READ ($46/$47)**

---

0048 **POINTER TO BLOCK READ ROUTINE**

004A **VOL DIR ENTRY POINTER/FIRST INDEX PAGE**

004B **ADDR OF SECOND PAGE OF INDEX BLOCK**

004D **INDEX INTO INDEX BLOCK PAGES**

0050 **TRACK SEEK PHASE-ON INDEX**

0051 **TRACK PHASE WANTED**

0052 **BLOCK READER RETRY COUNT**

0053 **CURRENT TRACK PHASE PHASE-OFF INDEX**

0060 **BUFFER POINTER**

0061 **SCREEN CENTER LINE**

2000 **LOAD POINT FOR RELOCATOR**

C880 **DISK ARM PHASES**

C888 **TURN DISK DRIVE OFF**

C889 **TURN DISK DRIVE ON**

C88C **SHIFT DATA REGISTER**

---

**FC59 HOME CURSOR/CLEAR SCREEN**

---

0600 **SIGNATURE BYTE ($01 MEANS BOOT ROUTINE FOLLOWS)**

(A $U3 IS STORED HERE DURING A 5.25" FLOPPY BOOT)

-- APPLE /// BOOTING --

THIS CODE (BLOCK 0) IS LOADED AT $A000 WHEN
BOOTED ON AN APPLE ///. THE APPLE /// BOOT
ROM JUMPS TO $A000. WHAT IS SHOWN HERE AS
$800 ON AN APPLE II IS $A000 ON AN APPLE ///.

THUS AN APPLE /// EXECUTES A HARMLESS
INSTRUCTION (ORA $38,X), THEN DOES NOT BRANCH
ON CARRY, AND JUMPS TO $A12 ($932 ON AN
APPLE II). MANY THANKS TO DAV HOLLE FOR
PROVIDING US WITH THIS APPLE /// INFORMATION.

---

0801 **MAIN ENTRY**

---

ON ENTRY, X = SLOT*16

A = SECTOR NUMBER

0801 **ENTRY POINT FOR APPLE II**

0802 **ALWAYS TAKEN (APPLE II) >>0807**

0804 **JUMP TO APPLE /// LOGIC >>A132**

0807 **SAVE SLOT*16**

0809 **READING SECTOR 3 NEXT?**

080B **REMEMBER THIS...**

080D **MARK $CX FROM SLOT*16**

0815 **AND SAVE AT $49**

0819 **$48/49 -- $CF PPP IN ROM BOOT**

081C **CHECK $CXPPP**

081D **BOOT ROM FOR DISK 117**

081F **NO, NOT A 5.25" FLOPPY >>085B**

0821 **GOT BOTH SECTORS OF LOADER? >>0831**

0823 **NO, STOP AT SECTOR 3**

0825 **STORE ON PARM ($806)**

0828 **SKIP SECTOR 1 (GET SEC 2)**

082A **DUMMY UP $CX5C AS RETURN ADDRESS**

0830 **AND CALL ROM SECTOR READ SUBRNM**

---

**LOAD PRODOS**

---

(ENTIRE LOADER IN MEMORY NOW)

0831 **CURRENT TRACK IS ZERO**

0833 **$48/49 -- $CX00**

0817 **COPY A PORTION OF DISKETTE BOOT ROM**

0819 **TO MY BLOCK READER SUBROUTINE ($994)**

083D **FROM $9F7 TO $AE7**

0843 **MODIFY SOME BRANCHES IN THE COPIED CODE ($91D)**

0846 **TO SUIT MY ERROR HANDLING TASTES ($924)**

084C **AND COPY SECTOR READ SUBROUTINE EXIT CODE ($929)**
ProDOS Loader -- V1.1.1 -- 18 SEP 84

ADDR DESCRIPTION/CONTENTS

084F TO $A7F TO $A85 (0A7F)
0855 $48/49 --> DISKETTE BLOCK READER SUBRNT
0859 AT $8906
085B --
085D LEGAL DISK ROM?
085F NO, ERROR >>0890
0861 STORE LSB OF BLOCK READER
0863 STORE ZEROS IN SEVERAL THINGS
0867 COMMAND = 1 (READ BLOCK)
0867 BLACK NUMBER = 2 (VOL DIRECTORY)
0875 $60/61 --> $C000 (BUFFER)
0877 $4A/4B --> $C00 (FIRST ENTRY)
0879 READ VOLUME DIRECTORY BLOCKS <0912>
087C ERROR? >>0890
087E MOVE UP TWO PAGES IN MEMORY
0882 NEXT BLOCK NUMBER
0886 NOW AT BLOCK 6?
0888 NJ, GO READ NEXT ONE >>0879
088A YES, CHECK LINK FOR VALIDITY (0C00)
088D IT SHOULD BE ZERO FOR VOL DIR (0C11)
088F BAD VOLUME DIR IF NOT ZERO >>08F0
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
08AA ELSE, START JUST PAST LINKS
08AC UPDATE LSB OF ENTRY POINTER
08AE GET NAME LENGTH (0902)
08B1 MASK OFF STORAGE TYPE
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
08C8 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 $60/61 --> $1E80
08DB (BUFFER TO HOLD KEY BLOCK)
08E1 $4C/4D --> $1F80 (SECOND PAGE)
08E3 READ A BLOCK <0912>
08E6 ERROR? >>08FF
08EA BUMP TO NEXT BLOCK BUFFER

ProDOS Loader -- V1.1.1 -- 18 SEP 84

ADDR DESCRIPTION/CONTENTS

08EE $4E = OFFSET INTO INDEX BLOCK
08F0 GET NEXT BLOCK NUMBER FROM INDEX BLOCK
08F8 BLOCK NUMBER = 0? (END OF FILE)
08FA NOT YET, READ A BLOCK >>08E3
08FC ELSE, JUMP TO RELOCATOR AT $2000 >>2000
08FF ERROR JUMP >>093F

0902 KERNEL NAME **************************************
0902 LENGTH OF KERNEL'S NAME
0903 'PRODOS' 'PRODOS' (KERNEL NAME)
0912 COPY BLOCK READ BUFFER PTR ************************
0912 COPY $60/61 --> $44/45
0914 (BLOCK READ BUFFER POINTER)
091A THEN GO TO BLOCK 1/0 ROUTINE >>0048
091D ROM SECTOR READ OFFSETS ****************************
091D OFFSETS INTO ROM SECTOR READ SUBROUTINE
091D TO BRANCH DISPLACEMENTS WHICH NEED TO
091D BE CHANGED FOR LOADER'S PURPOSES
091D ---
0920 NEW BRANCH OFFSETS FOR ABOVE ***

0924 ---

092B SECTOR READ EXIT CODE ****************************
092B COPYED TO END OF DISKETTE SECTOR READ CODE
092B GET SLOT*16
092D AND EXIT NORMALY
092E RETURN
092F RESTART BLOCK READ OPERATION >>099C

0932 APPLE /// BOOT CODE ******************************

A132 THIS IS $A132 WHEN BOOTED ON APPLE ///
0932 MAKE IT LOOK LIKE A JSR FROM $A000
093B LOAD IN BLOCK 1 (WE WANT SOS, NOT PRODOS)
093C GO TO APPLE /// BLOCK READ ROUTINE >>9479
093F ERROR HANDLER *******************************
Beneath Apple ProDOS Supplement

ProDOS Loader -- V1.1.1 -- 18 SEP 84
NEXT OBJECT ADDR: 093F

ADDR DESCRIPTION/CONTENTS
--------- -------------------------------
093F HOME CURSOR/CLEAR SCREEN <PC58> 0944 COPY "UNABLE TO LOAD PRODOS" MESSAGE (0950)
0947 TO SCREEN (0955)
094D THEN GO TO SLEEP FOREVER >>094D
0950 ---
0950 '*** UNABLE TO LOAD PRODOS ***'
0960 ********** MOVE ARM TO NEXT PHASE **********
096D GET CURRENT PHASE
096F CONVERT TO NEXT ARM PHASE
0972 ADD SLOT*16
0975 SELECT NEXT ARM PHASE THIS DRIVE (C060)
097A ---
097C DELAY LONG ENOUGH FOR ARM TO MOVE
096F WHEN FINISHED, RETURN WITH X = SLOT*16
0985 RETURN

0986 *********** DISKETTE BLOCK READ ROUTINE ***********
44/45 --> BUFFER
46/47 = BLOCK NO.
0986 GET BLOCK NO. LSB
0988 ISOLATE SECTOR REMAINDER
098C SKEW SECTOR BY 2
0992 AND STORE SECTOR WANTED
0994 GET MSB
0996 AND HIGH BIT OF TRACK
0999 MERGE WITH LOW PART OF TRACK
099C STORE TRACK WANTED
099F TRACK*2 IS PHASE WANTED
09A3 SET PAGE ADDRESS OF BUFFER
09A7 TURN DRIVE MOTOR ON (C089)
09AA READ SECTOR <98C>
09AD NEXT PAGE
09B1 SKEW TO NEXT SECTOR
09B5 READ SECOND SECTOR OF BLOCK <98C>
09BB THEN TURN MOTOR OFF AND EXIT (C086)
09B8 RETURN

******** DISKETTE SECTOR READ ROUTINE ***
09BC GET CURRENT TRACK
09BF CONVERT TO PHASE
09C5 GET CURRENT PHASE
09C7 STORE FOR PHASE OFF
09CA SUBTRACT PHASE WANTED TO DETERMINE...
09CC DIRECTION -- ON CORRECT TRACK NOW? >>09E2
09D0 NO, ADJUST PHASE UP...

ProDOS Loader -- V1.1.1 -- 18 SEP 84
NEXT OBJECT ADDR: 09D4

ADDR DESCRIPTION/CONTENTS
--------- -------------------------------
09D4 OR DOWN AND...
09D6 ---
09D7 SEEK ARM ONE PHASE... <96D>
09D9 IN PROPER DIRECTION <96F>
09DC UNTIL WE ARE THERE >>09C5
09E2 ---
09E4 RETRY COUNT OF 127
09E7 ---
09EB LOWER RETRY COUNT
09EC RETRIES EXHAUSTED? >>9BB
09F2 RETRIES FOR A $85 HEADER
09F2 CHECK DATA REGISTER (C08C)
09F5 LOOP UNTIL DATA IS VALID >>9F2

********** SECTOR READ ROUTINES **********
09F7 BEGINNING OF COPIED ROUTINES
(SEE $C65E IN BOOT FIRMWARE DESCRIPTIONS)
($C653-$C65A IS COPIED TO $9F7-$9E5)
0A7F EXIT CODE FOR READ ROUTINES
(COPIED HERE FROM $92B-$930)
0A86 ********** $A86-$BFF NOT USED **********
0A86 NOT USED
0C06 ********** VOLUME DIRECTORY BUFFER **********
0C00 START OF VOLUME DIRECTORY BUFFER
0C23 OFFSET TO ENTRY LENGTH FIELD
Beneath Apple ProDOS Supplement

ProDOS Relocator -- V1.1.1 -- 18 SEP 84  NEXT OBJECT ADDR: 2000

2000  MODULE STARTING ADDRESS

                      ***********************************************
                      *                                                *
                      *                 PRODOS RELOCATOR                *
                      *            LOADED AS THE FIRST               *
                      *           PORTION OF THE PRODOS              *
                      *            IMAGE AT $2000.                  *
                      *                                                *
                      *      VERSION 1.1.1 -- 18 SEP 84              *
                      *                                                *
                      ***********************************************

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

0488  SCREEN BUFFER LINE
05A9  SCREEN BUFFER LINE
05B1  SCREEN BUFFER LINE
06A8  SCREEN BUFFER LINE
07A8  SCREEN BUFFER LINE
07A9  SCREEN BUFFER LINE
07D0  SCREEN BUFFER LINE

                      ********** INTERP LOADER ADDRESSES **********

0800  ENTRY OF INTERP LOADER
08E2  'UNABLE TO FIND SYSTEM FILE'
090A  'INTERP FILE TOO LARGE'
092A  'UNABLE TO LOAD ...'
093B  INTERP FILE NAME ITSELF
093C  +1
094F  LENGTH OF MESSAGE
095B  MLI: OPEN LIST
095C  MLI: GET EOF
0958  EOF MARK
0959  EOF MARK+1
095A  EOF MARK+2 (MSB)
095B  MLI: READ LIST
095F  READ BUFFER ADDR
0960  +1
0963  MLI: CLOSE LIST
0965  '.SYSTEM'

                      ********** MISCELLANEOUS ADDRESSES **********

0C00  VOLUME DIRECTORY BUFFER
0C23  ENTRY LENGTH
       -- RAMDRIVE VOLUME DIRECTORY --
0E04  VOLUME HDR, VOLUME NAME
0E22  VOLUME HDR, ACCESS-TOTAL BLOCKS
2A00   DIFFERENCE OF RAMDRIVE LOAD AND RUN LOCATIONS
2B00   DIFFERENCE OF RAMDRIVE LOAD AND RUN LOCATIONS

                      ********** SYSTEM GLOBAL PAGE **********

BF00  ENTRY POINT FOR MLI
BF02  QUIT VECTOR
BF06  DATE/TIME
BF10  DEVICE HANDLER TABLES
BF30  LAST DEVICE USED
BF31  NUMBER OF ACTIVE DISK DEVICES
BF32  ACTIVE DISKS SEARCH LIST
BF98  MACHINE TYPE FLAGS
<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>BF99</td>
<td>SLOT WHICH CONTAIN CARDS WITH ROM</td>
</tr>
<tr>
<td>BF9F</td>
<td>TOP OF 40K RAM</td>
</tr>
<tr>
<td>C800</td>
<td>80 STORE OFF</td>
</tr>
<tr>
<td>C801</td>
<td>80 STORE ON</td>
</tr>
<tr>
<td>C802</td>
<td>READ MAIN RAM</td>
</tr>
<tr>
<td>C803</td>
<td>READ AUX RAM</td>
</tr>
<tr>
<td>C804</td>
<td>WRITE MAIN RAM</td>
</tr>
<tr>
<td>C805</td>
<td>WRITE AUX RAM</td>
</tr>
<tr>
<td>C808</td>
<td>MAIN STACK/ZERO PAGE</td>
</tr>
<tr>
<td>C809</td>
<td>ALTERNATE STACK/ZERO PAGE</td>
</tr>
<tr>
<td>C80A</td>
<td>INTERNAL SLOT 1 ROM</td>
</tr>
<tr>
<td>C80B</td>
<td>PERIPHERAL SLOT 1 ROM</td>
</tr>
<tr>
<td>C80C</td>
<td>80 COLUMN DISPLAY OFF</td>
</tr>
<tr>
<td>C80D</td>
<td>80 COLUMN DISPLAY ON</td>
</tr>
<tr>
<td>C818</td>
<td>READ BUS STORE SWITCH</td>
</tr>
<tr>
<td>C830</td>
<td>SPEAKER</td>
</tr>
<tr>
<td>C854</td>
<td>USE MAIN MEMORY PART OF 80-10 CARD</td>
</tr>
<tr>
<td>C855</td>
<td>USE AUX MEMORY PART OF 80-10 CARD</td>
</tr>
<tr>
<td>C851</td>
<td>WRITE-ENABLE HIGH RAM</td>
</tr>
<tr>
<td>C852</td>
<td>MOTHERBOARD ROM READ ENABLE</td>
</tr>
<tr>
<td>C853</td>
<td>READ/WRITE RAM 2ND 4K BANK</td>
</tr>
<tr>
<td>C858</td>
<td>READ/WRITE RAM 1ST 4K BANK</td>
</tr>
<tr>
<td>C311</td>
<td>MOVE TO/FROM AUXMEM SUBROUTINE</td>
</tr>
<tr>
<td>C314</td>
<td>TRANSFER TO/FROM AUXMEM SUBROUTINE</td>
</tr>
<tr>
<td>C305</td>
<td>SLOT3 I.D. BYTE</td>
</tr>
<tr>
<td>C307</td>
<td>SLOT3 I.D. BYTE</td>
</tr>
<tr>
<td>C308</td>
<td>SLOT3 I.D. BYTE</td>
</tr>
<tr>
<td>C30C</td>
<td>SLOT3 I.D. BYTE</td>
</tr>
<tr>
<td>C3FA</td>
<td>SLOT3 I.D. BYTE</td>
</tr>
<tr>
<td>CFFF</td>
<td>RESET I/O CARD ROMS</td>
</tr>
<tr>
<td>D800</td>
<td>START OF QUITCODE MEMORY AREA (BANK 2)</td>
</tr>
<tr>
<td>DFD8</td>
<td>ENHANCED ROM FLAG</td>
</tr>
<tr>
<td>FF00</td>
<td>RAMDRIVE CALLER ADDRESS</td>
</tr>
<tr>
<td>D800</td>
<td>START OF QUITCODE MEMORY AREA (BANK 2)</td>
</tr>
</tbody>
</table>

---

**ProDOS Relocator -- V1.1.1 -- 18 SEP 84**

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>FB1E</td>
<td>PADDLE READ SUBROUTINE</td>
</tr>
<tr>
<td>FB2F</td>
<td>MONITOR INIT ROUTINE</td>
</tr>
<tr>
<td>FBB3</td>
<td>ROM VERSION BYTE</td>
</tr>
<tr>
<td>FBCC</td>
<td>SECONDARY VERSION BYTE (0-3)</td>
</tr>
<tr>
<td>FC58</td>
<td>CLEAR SCREEN</td>
</tr>
<tr>
<td>FEB4</td>
<td>SET NORMAL VIDEO</td>
</tr>
<tr>
<td>FEB9</td>
<td>INI 0</td>
</tr>
<tr>
<td>FEB3</td>
<td>FR 0</td>
</tr>
</tbody>
</table>

---

**2000W **

********** PRODOS RELOCATOR MAIN ENTRY **********

---

**ProDOS Relocator -- V1.1.1 -- 18 SEP 84**

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td>STORE SLOT IN MLI ONLINE PARMS</td>
</tr>
<tr>
<td>2005</td>
<td>PRINT &quot;APPLE II PRODOS...&quot; &lt;2499&gt;</td>
</tr>
<tr>
<td>2006</td>
<td>SET UP FOR COMMON MOVES (220A)</td>
</tr>
<tr>
<td>2005</td>
<td>RELOCATE SOME ROUTINES &amp; DATA TO LOW MEMORY &lt;26A6&gt;</td>
</tr>
<tr>
<td>2011</td>
<td>ERROR? &gt;2836</td>
</tr>
<tr>
<td>2013</td>
<td>NO, PROCEED...</td>
</tr>
<tr>
<td>2017</td>
<td>BE SURE 4K OF MAIN MEMORY EXISTS (BFF)</td>
</tr>
<tr>
<td>2018</td>
<td>IF NOT, ERROR &gt;285E</td>
</tr>
<tr>
<td>2026</td>
<td>MAKE DOUBLY SURE &gt;208E</td>
</tr>
<tr>
<td>2028</td>
<td>SELECT MOTHERBOARD ROMS (C852)</td>
</tr>
<tr>
<td>2023</td>
<td>DETERMINE MACHINE TYPE &lt;2402&gt;</td>
</tr>
<tr>
<td>2038</td>
<td>PICK UP CONFIGURATION BYTE</td>
</tr>
<tr>
<td>2032</td>
<td>64K OR MORE MEMORY?</td>
</tr>
<tr>
<td>2034</td>
<td>YES, WE HAVE 64K RAM &gt;2839</td>
</tr>
<tr>
<td>2036</td>
<td>ERROR. MUST HAVE 64K FOR PRODOS 1.1.11:1 &gt;21C3</td>
</tr>
</tbody>
</table>

---

**RELOCATE PRODOS **********

---

**ProDOS Relocator -- V1.1.1 -- 18 SEP 84**

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>2039</td>
<td>SET UP FOR MLI MOVE (220C)</td>
</tr>
<tr>
<td>203F</td>
<td>COPY/RELOCATE PRODOS ITSELF &lt;26A6&gt;</td>
</tr>
<tr>
<td>2042</td>
<td>ERROR? &gt;285E</td>
</tr>
<tr>
<td>2044</td>
<td>ENABLE MOTHERBOARD ROMS AGAIN (C852)</td>
</tr>
<tr>
<td>2047</td>
<td>CHECK ROM I.D. BYTE (FBB3)</td>
</tr>
<tr>
<td>204A</td>
<td>APPLE //e FAMILY?</td>
</tr>
<tr>
<td>204C</td>
<td>NO, LEAVE I.D. BYTE AS IS &gt;206D</td>
</tr>
<tr>
<td>2059</td>
<td>TEST ANOTHER ROM I.D. BYTE (FBCC)</td>
</tr>
<tr>
<td>2053</td>
<td>SAVE BIT TEST RESULTS</td>
</tr>
<tr>
<td>2054</td>
<td>GET MACHID</td>
</tr>
<tr>
<td>2056</td>
<td>STRIP BITS THAT IDENTIFY MODEL</td>
</tr>
<tr>
<td>205B</td>
<td>IT'S A //e IF BITS 6 &amp; 7 ARE HIGH &gt;2069</td>
</tr>
<tr>
<td>205D</td>
<td>---</td>
</tr>
<tr>
<td>205E</td>
<td>EITHER A //c OR A FUTURE SYSTEM</td>
</tr>
<tr>
<td>2066</td>
<td>CHECK HIGH BITS OF $FBCC AGAIN</td>
</tr>
<tr>
<td>2061</td>
<td>BIT 7 ON? &gt;2067</td>
</tr>
<tr>
<td>2063</td>
<td>YES, FUTURE SYSTEM.</td>
</tr>
<tr>
<td>2067</td>
<td>IF BIT 6 ON, IT'S A FUTURE SYSTEM. &gt;206B</td>
</tr>
<tr>
<td>2069</td>
<td>---</td>
</tr>
<tr>
<td>2068</td>
<td>REPLACE UPDATED MACHID</td>
</tr>
<tr>
<td>206D</td>
<td>COPY BOOT DEVICE ID TO READ BLOCK PARMS (21FE)</td>
</tr>
</tbody>
</table>
ProDOS Relocator -- V1.1.1 -- 18 SEP 84  NEXT OBJECT ADDR: 2073

ADDRES DESCRIPTION/CONTENTS

2073 AND AS LAST DEVICE USED (BF3F)
2076 DETERMINE PERIPHERAL CARD CONFIGURATION <252A>
2079 BOOT DEVICE TO... (2205)
207C GLOBAL PAGE LAST DEVICE USED (BF3F)
2082 WRITE ENABLE BANK1 OF HIGH RAM (C088)
208B COPY CLOCK CODE TO DEVICE DRIVER AREA <26A6>
208E ERROR? >>2BF
2090 CHECK MACHINE TYPE AGAIN (BF98)
2093 GOT 64K OR MORE?
2097 NO >>20C2
2099 YES, QUIT VECTOR --> $FCE5
20A3 WRITE TO HIGH RAM (BANK2) (C083)
20AC POINT TO QUIT CODE TABLE (2111)
20AF MOVE QUIT CODE TO HIGH RAM <26A6>
20B4 STORE QUIT VECTOR START PAGE (D000)
20B7 WRITE TO HIGH RAM (BANK1) (C088)
20BA AGAIN (C08B)
20BF RELOCATION ERROR >>21C3
20C2 GET MACHINE YET AGAIN (BF98)
20C5 128K?
20C9 NO... >>20D1
20CC YES, ESTABLISH RAM DRIVE IN UPPER 64K <2FF>

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

20D1 READ ROM (C081)
20D4 GET IRQ VECTOR FROM ROM (FFE)
20DA CARRY CLEAR IF IRQ VECTOR IN C3 ROM
20DD IT'S AN OLD ROM >>26FD
20DE READ & WRITE RAM (BANK1) (C08B)
20E5 SWITCH TO AUX HIGH RAM (C089)
20EB PUT IRQ VECTOR IN AUX HIGH RAM (FFFF)
20EF BACK TO MAIN HIGH RAM, 2-PAGE (C086)
20F2 PUT IRQ VECTOR IN MAIN HIGH RAM (FFFF)
20F9 SET FLAG INDICATING
20FA ENHANCED IRQ LOGIC ON BOARD (DFD8)

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

20FD ENABLE INTERNAL VIDEO FIRMWARE (C08A)
2100 CHECK FOR ROM (BF99)
2103 IN SLOT 3.
2105 NONE THERE >>216D
2107 LOOK AT THE SLOT 3 ROM (C08B)
210A AT OFFSET +$85 (C085)
2110 THERE MUST BE A $38
2111 AND AT OFFSET +$07 (C307)
2114 THERE MUST BE A $18
2118 AND AT OFFSET +$09 (C08B)
211B THERE MUST BE A L
211F AND AT OFFSET +$8C (C30C)

ProDOS Relocator -- V1.1.1 -- 18 SEP 84  NEXT OBJECT ADDR: 2124

ADDRES DESCRIPTION/CONTENTS

2124 INDICATE AN 80-COL CARD.
2128 CHECK MACHINE TYPE (BF96)
212D IS THIS AN APPLE III?
212F OK, IT'S GOT 80-COL CAPABILITY >>2165
2131 OTHER MANUFACTURERS MUST FOLLOW THE RULES! (C3FA)
2134 MUST HAVE BIT INSTRUCTION AT $C3FA
2136 GOOD JOB, YOU FOLLOWED THE RULES! >>2165
2138 GIVE CONTROL BACK TO MOTHERBOARD ROM (C06A)
213B TURN ON 80-COL (C061)
213E CHECK FOR AUX MEM. (C055)
2143 PUT A BYTE AT AUX $400 (0408)
2146 THE ACCUMULATOR LEFT
2147 AND THE SAME WITH $400 (0408)
214A STILL THE SAME? (0408)
214D NO, NO 80-COL MEMORY >>2156
214F SHIFT TO THE RIGHT
2153 STILL THE SAME? (0408)
2156 BACK TO MAIN MEMORY (C054)
2159 TURN OFF 80-COL (C060)
215C WAS 80-COL MEMORY FOUND? >>2165
215E NO, SO TURN OFF 80-COL FLAG (BF98)
2161 IN MACHINE I.D. BYTE.
2163 ALWAYS BRANCH >>216A
2165 TURN ON 80-COL FLAG (BF98)

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

216D ML1: ONLINE DEVICE CALL <BF00>
2173 ERROR? >>21C3
2178 VALID VOLUME NAME?
217A IF NOT, ERROR >>21C3
217D ELSE, BUMP LENGTH BY ONE
21B2 AND PREFIX NAME BY A "/
21B7 ML1: SET PREFIX <BF00>
21BD ERROR? >>21C3

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

21BF ---
2191 $14/15 --> $C00
2197 ---
219C BLOCK = 2 (VOLUME DIRECTORY) (2208)
21A2 ML1: READ BLOCK <BF00>
21AB ERROR? >>21C3
21AC GET NEXT BLOCK NUMBER
21B2 IF ZERO, END OF VOLUME DIRECTORY >>2C8
21BA ADD TWO PAGES (ONE BLOCK) TO POINTER
21BC AND STOP AT $1400 IN ANY CASE
21BE ELSE, READ NEXT BLOCK AS WELL >>2197
21C8 WHEN DONE, JUMP TO SYSTEM FILE LOADER >>0800
ProDOS Relocator — V1.1.1 — 18 SEP 84  
NEXT OBJECT ADDR: 21C0

ADDR DESCRIPTION/CONTENTS

21C3 ************ ERROR HANDLER ********************
21C3 ENABLE MOTHERBOARD ROMS (C802)
21C6 CLEAR SCREEN <FC58>
21CB PRINT "RELOCATION/CONFIG ERROR" (21D7)
21D4 THEN SLEEP FOREVER >>21D4

21D7 ************ DATA ***********************

21D7 ---
21D7 *** RELOCATION / CONFIGURATION ERROR ***

21FD ML1: ONLINE PARMS
21FE SLOT*16 AND DRIVE
21FF READ THEM TO $281

2201 ML1: SET PREFIX PARMS
2202 PREFIX IS AT $280

2204 ML1: READ BLOCK PARMS
2205 DEVICE
2206 BUFFER
2208 BLOCK NUMBER

220A ADDRESS OF COMMON MOVES RELOC TABLE
220C ADDRESS OF PRODOS RELOC TABLE
220E ADDRESS OF CLOCK DRIVER RELOC TABLE
2210 ADDRESS OF QUIT CODE RELOC TABLE

2212 *********** RELLOCATION TABLES ***********************

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

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

2212 COPY (SYSTEM FILE LOADER)
2213 TO =$800
2215 LEN=$16C
2217 FRM=$226C
2219 COPY (PAGE 3 IMAGE)
Beneath Apple ProDOS Supplement

ProDOS Relocator -- V1.1.1 -- 18 SEP 84

ADDR DESCRIPTION/CONTENTS

2261 TO =$D742
2263 LEN= $69
2265 FRM= $D742
2267 FOR ADDR= $C1XX-$C1XX
226A ADJUST BY= $S0
226B END OF TABLE

226C ********** SYSTEM FILE LOADER ******************************************
(COPIED TO AND RUN AT $E900)

226C $10/11 --> VOLUME DIRECTORY ENTRIES
226E INITIALLY AT $S00
2270 OFFSET BEYOND LINKS (+4)
2272 (TURN NEXT INSTRUCTION INTO BIT)

******** SCAN DIRECTORY FOR SYSTEM FILE *

2273 PICK UP LSB
2276 BUMP BY ENTRY LENGTH ($C23)
2279 UPDATE LSB
227B PAGE OVERFLOW? >>228F
227D NO, BOOM FOR ONE MORE ENTRY? ($C23)
2282 NO, CHECK MSB
2285 START OF A BLOCK? >>2291
2287 NO, AT END OF DIRECTORY?
228B YES, FILE NOT FOUND IN DIRECTORY >>22A9
228B NO, START NEW BLOCK AT +4
228D AND UPDATE LSB
228F BUMP MSB
2291 ---
2295 CHECK FILE TYPE FOR PRODOS "SYS" FILE
2297 NOT IT? >>2273
2299 INACTIVE ENTRY?
229C IF SO, SKIP IT >>2273
22A0 SAVE NAME LENGTH AT $280 ($828)
22A5 MUST BE AT LEAST 8 CHARLS LONG >>2273
22A7 JUMP AROUND ERROR CODE >>22AC
22A9 ERROR - SYSTEM FILE NOT FOUND >>2319
22AB HARD BREAK IN THAT CASE
22AC ---
22AF IS THIS ".SYSTEM"?
22B1 (SEE $23D1) ($965)
22B5 NO, SKIP Entry >>2273
22B9 CHECK ALL CHARACTERS IN NAME >>22AF

******* LOAD SYSTEM FILE AT $2000 *****
ProDOS Relocator — V1.1.1 — 19 SEP 84

ADDR  DESCRIPTION/CONTENTS

2340  ---
2342  PRINT "SYSTEM PROGRAM TOO LARGE" (E90A)
234B  GO TO SLEEP FOREVER >>2348

234E  ********* DATA AREA *************

234E  "" UNABLE TO FIND A ".SYSTEM" FILE ""
2376  "" SYSTEM PROGRAM TOO LARGE ""
2396  "" UNABLE TO LOAD X.SYSTEM ""
23BB  NAME LEN +13H (LEN OF MSG)

23BC  MLI: OPEN PARM LIST
23BD  PATHNAME IS AT $280
23BF  I/O BUFFER AT $1400
23C1  REFNUM=1

23C2  MLI: GET EOF PARM LIST
23C3  REFNUM=1
23C4  EOF MARK POSITION

23C7  MLI: READ LIST
23C8  REFNUM=1
23C9  READ TO $2000
23CB  LENGTH (FROM EOF MARK)
23CD  ACTUAL LENGTH READ

23CF  MLI: CLOSE LIST
23D0  REFNUM=8, CLOSE ALL FILES
23D1  ".SYSTEM"

23D8  ********** END OF SYSTEM FILE LOADER *************

23D8  ********** PAGE 3 VECTOR IMAGE **************

(INCLUDES A ROUTINE AT $3D6 THAT COPIES
CRITICAL ZERO PAGE VALUES TO AUX MEM)

23D8  FROM MAIN Z-PAGE, (C000)
23DB  GET X1 VALUES STARTING AT $42
23D0  AND PUT IN AUX Z-PAGE (C009)
23E0  AT SAME LOCATION.
23E5  "NO DEVICE CONNECTED" ERROR
23E6  BACK TO MAIN Z-PAGE (C000)
23E8  RETURN
23EC  ADDRESS OF MLI ROUTINE
23F2  BRK HANDLER AT $FA59
23F4  RESET AT $FF59
23F6  POWER UP BYTE
23F7  & VECTOR TO $FF59 >>FF59

ProDOS Relocator — V1.1.1 — 19 SEP 84

ADDR  DESCRIPTION/CONTENTS

23FA  CTL-Y VECTOR TO $FF59 >>FF59
23FD  INT VECTOR TO $FF59 >>FF59
2400  IRQ HANDLER AT $BE00 (PRODOS)

2402  ********** DETERMINE MACHINE ID **********

$8C=00: 0... APPLE II
01... 0... APPLE IIa
10... 1... APPLE IIc
11... 0... APPLE /// EMULAT.
... 01... 48K RAM
... 10... 64K RAM
... 11... 128K RAM
... 12... 80 COL CARD
... 13... THUNDER CLOCK

2402  ASSUME NOTHING AT FIRST
2406  GET A ROM BYTE (F883)
2409  APPLE II?
240B  YES, SET BIT >>24E
240D  NO,
240F  APPLE IIk?
2411  YES, SET BIT >>24E
2413  NO,
2415  APPLE II+?
2417  NO, WHAT IS IT? >>242B
241C  REALLY A II+?
241E  YES >>24E
2422  /// EMULATION MODE?
2426  ---
2427  RETURN
2428  OTHERWISE, UNKNOWN MACHINE
242A  CREATE INVALID INSTR AT $80
242C  AND GO THERE >>244E
242E  UPDATE MACHID
2433  READ/WRITE ENABLE HIGH RAM (BANK1) (C00B)
2438  SEE IF HIGH RAM EXISTS (D000)
244A  IF PRESENT, MARK IN MACHID

2451  ********** LOOK FOR 64K OF AUX RAM **********

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

2451  UPDATE MACHID
2453  II17 >>248A
2455  YES,
2457  BANK TO AUX MEMORY (C005)
245D  STORE A PATTERN AT $C000 (8C00)
2460  AND AT $8000 (0800)
2466  MAKE SURE PATTERN STAYS THERE
2468  IT DIDN'T! >>2478
ProDOS Relocator -- V1.1.1 -- 18 SEP 84

246A NOW SHIFT $C00 TO THE LEFT (0C00)
246D AND SHIFT THE ACCUM TO THE LEFT
246E ARE THEY STILL THE SAME? (0C00)
2471 NO, AUX RAM NOT THERE. >>247B
2473 DID $800 MOVE TOO? (8800)
2476 NO, SO WE HAVE FULL 128K! >>247B
247B DON'T HAVE 128K
247B ---
247C BANK BACK TO MAIN MEMORY (C004)
2482 64K? >>248A
2486 NO, INDICATE 128K
2488 IN MACHID
248A SET UP $A/B --> "APPLE II"
248D IN MOTHERBOARD ROM
248F AT $FB09
2491 BUT DO IT IN A CONVOLUTED WAY
2498 RETURN TO CALLER

2499 ********** DISPLAY LOAD MESSAGE ****************************

2499 CLICK SPEAKER (C039)
249C STORE IN MAIN MEMORY (C00C)
249F 00 COL DISPLAY OFF (C000)
24A2 SET NORMAL VIDEO <FB84>
24A5 CALL MONITOR INITIALIZATION <FB2F>
24A8 SET VIDEO PR0 <FB93>
24AB SET KEYED M0 <FB89>
24AE OUT OF DECIMAL MODE
24AF DISABLE FOR INTERRUPTS
24B8 CLEAR SCREEN <FC5B>
24B5 PRINT "APPLE //" (24E3)"
24C0 PRINT "PRODOS 1.1.1 ETC." (24E8)
24CB PRINT A BLANK AT 6A8 (2502)
24D6 PRINT "COPYRIGHT ETC." (2503)
24DF CLICK SPEAKER AGAIN (C039)
24E2 DONE

24E3 ********** DATA AREA *****************************************

24E3 'APPLE //'
24EB 'PRODOS 1.1.1 18-SEP-84'
2502 '
2503 'COPYRIGHT APPLE COMPUTER, INC., 1983-84'

25A2 ********** DETERMINE SLOT CONFIGURATION **********************

25A2 ---
25A2 ZERO SOME THINGS
25A3 NO DISKS ACTIVE YET (BF31)
25A8 $10/11 --> $C700 (LOOP THRU ALL SLOTS)
25A3A RESET 1/0 CARD ROMS (CFFF)

ProDOS Relocator -- V1.1.1 -- 18 SEP 84

253F CHECK SIGNATURE ON CARD FOR DISK DEVICE
2545 NOT DISK? >>25AD
254B GET $C0FF BYTE (TYPE OF DISK)
254D DISK II? >>256F
254F NO, PROFILE?
2551 NO? THEN NOT A DISK >>25AD

******* PROFILE FOUND ***************

2553 ELSE, SAVE AS LSB OF BLOCK READ SUBRTN
2555 GET $C8FE (STATUS BYTE)
255B CAN WE AT LEAST READ STATUS AND DATA?
255C YES? >>2563
255E NO,
2561 NOT A DISK AFTER ALL >>25AD
2563 GET STATUS BYTE AGAIN
2567 TOP NIBBLE IS DEVICE ID
256B PROFILE SHOULD BE $04
256A CHECK NUMBER OF VOLS (SHOULD BE 0)
256B GET SLOT NO. FOR DEVICE DRIVER LOCS.
256D AND GO DO COMMON PROCESSING FOR DISK >>2579

******* DISK II FOUND ***************

256F $12 ZERO FOR DISK II
2571 GET DISK II DEVICE DRIVER LOCATION (266A)
2575 ($FB00 OR $FB00) (266B)
257B DISK II HAS 2 DRIVES

******* DISK FOUND *******************

2579 SAVE DEVICE ADDRESS
257B SET UP INDEX OF SLOT#2
2593 BUILD ST (S=SLT, T=0 DISKII4 PROFILE)
2596 BUMP DEVICE COUNT BY ONE (BF31)
25BA AND ADD DRIVE TO SYSTEM SEARCH LIST (BFJ2)
25BE NUMBER OF DRIVES
2590 ONLY ONE? >>2596
2592 NO, BUMP INDEX
2593 AND MARK SECOND DRIVE IN SEARCH LIST (BF32)
2596 STORE FINAL DEVICE COUNT (BFJ1)
259B SET UP DISK DEVICE DRIVER VECTORS (BF11)
259E IN SYSTEM GLOBAL PAGE >>25EB
25A8 (SET UP TWO VECTORS FOR A DISK II) (BF21)
25AB ---
25AC I RECOGNIZE THIS CARD
25AD GO MARK SLTBYT TO SHOW ROMS IN SLOT <25FA>
25B4 DO ALL CARDS EXCEPT
25B6 SLOT 0 ($C000) >>253A
25B8 GET LAST DISK DEVICE IN SEARCH LIST (BF32)
25C2 BOOT DRIVE? (BF30)
<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>25C6</td>
<td>NO, KEEP LOOKING &gt;&gt;25CA</td>
</tr>
<tr>
<td>25D0</td>
<td>GET DEVICE COUNT (BF31)</td>
</tr>
<tr>
<td>25D2</td>
<td>IS BOOT DRIVE IN LIST? &gt;&gt;25E7</td>
</tr>
<tr>
<td>25D3</td>
<td>SO IT WILL BE SEARCHED FIRST... (BF49)</td>
</tr>
<tr>
<td>25D4</td>
<td>STORE BOOT AT END OF SEARCH LIST (BF32)</td>
</tr>
<tr>
<td>25D9</td>
<td>ANY OTHERS? &gt;&gt;25EE</td>
</tr>
<tr>
<td>25E8</td>
<td>YES, SECOND DRIVE? &gt;&gt;25E7</td>
</tr>
<tr>
<td>25E9</td>
<td>STORE IT RIGHT BEHIND BOOT DRIVE (BF32)</td>
</tr>
<tr>
<td>25EC</td>
<td>NOW ANY MORE? &gt;&gt;25EE</td>
</tr>
<tr>
<td>25F0</td>
<td>YES, MOVE OTHERS AHEAD IN LIST (BF32)</td>
</tr>
<tr>
<td>25F1</td>
<td>DO CHECKSUM ON ROM &lt;267C&gt;</td>
</tr>
<tr>
<td>25F2</td>
<td>NOT AN AUTOSTART ROM? &gt;&gt;25F7</td>
</tr>
<tr>
<td>25F3</td>
<td>AUTOSTART, STORE FINISHED MACHID (BF98)</td>
</tr>
<tr>
<td>25F4</td>
<td>AND LEAVE</td>
</tr>
<tr>
<td>25F5</td>
<td>NONAUTOSTART, UNKNOWN MACHINE, SO CRASH! &gt;&gt;2428</td>
</tr>
</tbody>
</table>

**25FA IDENTIFY I/O CARD**

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>25F6</td>
<td>DO WE ALREADY RECOGNIZE THIS CARD? &gt;&gt;265B</td>
</tr>
<tr>
<td>25F7</td>
<td>NO,</td>
</tr>
<tr>
<td>25F8</td>
<td>CHECK SIGNATURE ON CARD FOR THUNDER CLOCK</td>
</tr>
<tr>
<td>25FA</td>
<td>NOT IT? &gt;&gt;261F</td>
</tr>
<tr>
<td>25FB</td>
<td>THUNDER CLOCK, WHICH SLOT?</td>
</tr>
<tr>
<td>25FC</td>
<td>SAVE SLOT NUMBER (LESS 1)</td>
</tr>
<tr>
<td>25FD</td>
<td>IN CLOCK CODE RELOCATION TABLE (226A)</td>
</tr>
<tr>
<td>25FE</td>
<td>ENABLE CLOCK/CALENDAR JUMP IN GLOBALS (BF06)</td>
</tr>
<tr>
<td>2600</td>
<td>IS THERE A MACHID? &gt;&gt;25EE</td>
</tr>
<tr>
<td>2602</td>
<td>IF SO, MARK THAT A CLOCK IS PRESENT</td>
</tr>
<tr>
<td>2603</td>
<td>AND UPDATE MACHID</td>
</tr>
<tr>
<td>2605</td>
<td>GO MARK ROM IN THIS SLOT &gt;&gt;265B</td>
</tr>
<tr>
<td>2606</td>
<td>YES,</td>
</tr>
<tr>
<td>2607</td>
<td>CHECK SIGNATURE OF MYSTERY CARD</td>
</tr>
<tr>
<td>2608</td>
<td>STANDARD BASIC SUPPORTED?</td>
</tr>
<tr>
<td>2609</td>
<td>NO, UNKNOWN CARD &gt;&gt;264A</td>
</tr>
<tr>
<td>260A</td>
<td>YES,</td>
</tr>
<tr>
<td>260B</td>
<td>DOUBLE CHECK BASIC SUPPORTED</td>
</tr>
<tr>
<td>260C</td>
<td>NO, UNKNOWN CARD &gt;&gt;264A</td>
</tr>
<tr>
<td>260D</td>
<td>YES,</td>
</tr>
<tr>
<td>260E</td>
<td>GENERIC SIGNATURE</td>
</tr>
<tr>
<td>260F</td>
<td>NO, UNKNOWN CARD &gt;&gt;264A</td>
</tr>
<tr>
<td>2610</td>
<td>YES,</td>
</tr>
<tr>
<td>2611</td>
<td>B8 COLUMN CARD?</td>
</tr>
<tr>
<td>2612</td>
<td>NO, UNKNOWN CARD &gt;&gt;264A</td>
</tr>
<tr>
<td>2613</td>
<td>GET MACHID IF WE HAVE ONE &gt;&gt;25EE</td>
</tr>
<tr>
<td>2614</td>
<td>MARK B8 COLUMN CARD PRESENT</td>
</tr>
<tr>
<td>2615</td>
<td>AND UPDATE MACHID</td>
</tr>
<tr>
<td>2616</td>
<td>GO MARK ROM ON CARD PRESENT &gt;&gt;265B</td>
</tr>
<tr>
<td>2617</td>
<td>UNKNOWN CARD, CHECK ROM TO...</td>
</tr>
<tr>
<td>2618</td>
<td>SEE IF IT WILL HOLD A VALUE...</td>
</tr>
</tbody>
</table>
### ProDOS Relocator -- V1.1.1 -- 18 SEP 84

**NEXT OBJECT ADDR: 26E4**

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>26E4</td>
<td>COPY BLOCK ONLY? &gt;&gt;2753</td>
</tr>
<tr>
<td>26E5</td>
<td>SAVE RELOCATION OPERATION CODE (287F)</td>
</tr>
<tr>
<td>26E6</td>
<td>SAVE NUMBER OF RANGES TO CHECK (2880)</td>
</tr>
<tr>
<td>26F0</td>
<td>---</td>
</tr>
<tr>
<td>26F1</td>
<td>COPY START PAGES TO TABLE</td>
</tr>
<tr>
<td>26F2</td>
<td>---</td>
</tr>
<tr>
<td>26F3</td>
<td>AND END PAGES</td>
</tr>
<tr>
<td>26F9</td>
<td>AND FINALLY, RELOCATION FACTORS</td>
</tr>
<tr>
<td>2711</td>
<td>BUMP TO NEXT TABLE ENTRY &lt;2759&gt;</td>
</tr>
<tr>
<td>2714</td>
<td>RESTORE OPERATION CODE (287F)</td>
</tr>
<tr>
<td>2719</td>
<td>RELOCATE INSTRUCTIONS? &gt;&gt;2729</td>
</tr>
</tbody>
</table>

#### 271B ********** 2/3 - RELOCATE ADDRESSES ***********************

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>271B</td>
<td>NO, RELOCATE ADDRESS &lt;27BD&gt;</td>
</tr>
<tr>
<td>271E</td>
<td>COPY BLOCK &lt;2766&gt;</td>
</tr>
<tr>
<td>2721</td>
<td>AND CONTINUE IF ALL WENT WELL &gt;&gt;26AA</td>
</tr>
<tr>
<td>2724</td>
<td>NORMAL EXIT</td>
</tr>
<tr>
<td>2725</td>
<td>RETURN</td>
</tr>
<tr>
<td>2726</td>
<td>JUMP TO ERROR EXIT &gt;&gt;27F3</td>
</tr>
</tbody>
</table>

#### 2729 ********** 4 - RELOCATE INSTRUCTIONS ***********************

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>2729</td>
<td>RELOCATE INSTRUCTIONS &lt;27CF&gt;</td>
</tr>
<tr>
<td>272C</td>
<td>AND THEN COPY BLOCK &gt;&gt;271E</td>
</tr>
</tbody>
</table>

#### 272F ********** Ø - ZERO BLOCK ***********************

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>272F</td>
<td>BUMP TABLE POINTER TO NEXT ENTRY &lt;2759&gt;</td>
</tr>
<tr>
<td>2734</td>
<td>GET NUMBER OF PAGES TO DO</td>
</tr>
<tr>
<td>2736</td>
<td>NO FULL PAGES? &gt;&gt;2744</td>
</tr>
<tr>
<td>2739</td>
<td>ZERO AN ENTIRE PAGE</td>
</tr>
<tr>
<td>273E</td>
<td>BUMP PAGE POINTER</td>
</tr>
<tr>
<td>2740</td>
<td>AND DECREMENT LENGTH</td>
</tr>
<tr>
<td>2744</td>
<td>GET LENGTH OF PARTIAL LAST PAGE</td>
</tr>
<tr>
<td>2746</td>
<td>NO PARTIAL PAGE? &gt;&gt;2750</td>
</tr>
<tr>
<td>2749</td>
<td>ZERO PARTIAL PAGE TOO</td>
</tr>
<tr>
<td>2750</td>
<td>DONE, GET NEXT TABLE ENTRY &gt;&gt;26AA</td>
</tr>
</tbody>
</table>

#### 2753 ********** 1 - COPY BLOCK ***********************

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>2753</td>
<td>BUMP TABLE POINTER &lt;2759&gt;</td>
</tr>
<tr>
<td>2756</td>
<td>AND GO COPY BLOCK &gt;&gt;271E</td>
</tr>
</tbody>
</table>

#### 2759 ********** ADVANCE TABLE POINTER ***********************

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>2759</td>
<td>ADD FINAL ENTRY INDEX..</td>
</tr>
<tr>
<td>275D</td>
<td>TO TABLE ENTRY ADDRESS</td>
</tr>
<tr>
<td>2765</td>
<td>RETURN</td>
</tr>
</tbody>
</table>

#### 2766 ********** COPY BLOCK ***********************

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>2766</td>
<td>---</td>
</tr>
<tr>
<td>276A</td>
<td>INPTR &lt; OUTPTR? &gt;&gt;2777</td>
</tr>
<tr>
<td>276C</td>
<td>NO, GREATER? &gt;&gt;279A</td>
</tr>
<tr>
<td>276E</td>
<td>MSB'S ARE EQUAL, CHECK LSB'S ALSO</td>
</tr>
<tr>
<td>2776</td>
<td>EXIT IF EQUAL</td>
</tr>
<tr>
<td>2777</td>
<td>INPTR &lt; OUTPTR, COPY LAST PAGES FIRST</td>
</tr>
<tr>
<td>277B</td>
<td>BUMP BOTH INPTR AND OUTPTR BY...</td>
</tr>
<tr>
<td>277D</td>
<td>LENGTH-1 TO POINT AT LAST BYTE</td>
</tr>
<tr>
<td>2765</td>
<td>START WITH SHORT LAST PAGE LENGTH</td>
</tr>
<tr>
<td>2789</td>
<td>---</td>
</tr>
<tr>
<td>278A</td>
<td>COPY BYTES BACKWARDS THROUGH MEMORY</td>
</tr>
<tr>
<td>2791</td>
<td>DROP ADDRESSES AND LENGTH BY 256</td>
</tr>
<tr>
<td>2797</td>
<td>AND CONTINUE UNTIL FINISHED &gt;&gt;2789</td>
</tr>
<tr>
<td>2799</td>
<td>RETURN</td>
</tr>
</tbody>
</table>

#### 279A INPTR > OUTPTR, COPY PAGES FORWARD |

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>279C</td>
<td>HOW MANY FULL PAGES LEFT?</td>
</tr>
<tr>
<td>279E</td>
<td>NONE? &gt;&gt;27AF</td>
</tr>
<tr>
<td>27A0</td>
<td>COPY A FULL PAGE</td>
</tr>
<tr>
<td>27A7</td>
<td>AND BUMP ADDRESSES</td>
</tr>
<tr>
<td>27AB</td>
<td>DECREMENT LENGTH BY 256</td>
</tr>
<tr>
<td>27AD</td>
<td>AND DO ALL PAGES &gt;&gt;27AB</td>
</tr>
<tr>
<td>27AF</td>
<td>GET LENGTH OF LAST PAGE</td>
</tr>
<tr>
<td>27B1</td>
<td>EVEN PAGE BOUNDARY? &gt;&gt;27BC</td>
</tr>
<tr>
<td>27BC</td>
<td>NO, COPY SHORT LAST PAGE</td>
</tr>
<tr>
<td>27BC</td>
<td>RETURN</td>
</tr>
</tbody>
</table>

#### 27BD ADDR/PAGE RELOCATE ***********************

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>27BD</td>
<td>GET TABLE ENTRY TYPE (287F)</td>
</tr>
<tr>
<td>27C1</td>
<td>GET PAGE TO RELOCATE</td>
</tr>
<tr>
<td>27C3</td>
<td>RELocate A SINGLE ADDRESS &lt;27FB&gt;</td>
</tr>
<tr>
<td>27C6</td>
<td>BUMP BY 1 OR 2 BYTES (287F)</td>
</tr>
<tr>
<td>27C9</td>
<td>ADVANCE POINTER &lt;2817&gt;</td>
</tr>
<tr>
<td>27CC</td>
<td>AND CONTINUE UNTIL COMPLETE &gt;&gt;27BD</td>
</tr>
<tr>
<td>27CE</td>
<td>RETURN</td>
</tr>
</tbody>
</table>

#### 27CF INSTRUCTIONS RELOCATE ***********************

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>27CF</td>
<td>---</td>
</tr>
<tr>
<td>27D1</td>
<td>GET 6502 OPCODE</td>
</tr>
<tr>
<td>27D3</td>
<td>COMPUTE INSTRUCTION LENGTH &lt;282A</td>
</tr>
<tr>
<td>27D6</td>
<td>INVALID OPCODE? &gt;&gt;27E9</td>
</tr>
<tr>
<td>27DB</td>
<td>3 BYTE INSTRUCTIONS?</td>
</tr>
</tbody>
</table>
Beneath Apple ProDOS Supplement

ProDOS Relocator -- V1.1.1 -- 18 SEP 84  NEXT OBJECT ADDR: 27DA

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>27DA</td>
<td>NO &gt;&gt;27E3</td>
</tr>
<tr>
<td>27DC</td>
<td>YES, 3 BYTE ADDRESS TO CORRECT</td>
</tr>
<tr>
<td>27DE</td>
<td>RELOCATE ADDRESS &lt;&lt;27F&gt;</td>
</tr>
<tr>
<td>27EL</td>
<td>AND ADVANCE BY 3 BYTES</td>
</tr>
<tr>
<td>27E3</td>
<td>NEXT INSTRUCTION &lt;&lt;2817&gt;</td>
</tr>
<tr>
<td>27E6</td>
<td>CONTINUE UNTIL FINISHED &gt;&gt;27CF</td>
</tr>
<tr>
<td>27EB</td>
<td>RETURN</td>
</tr>
</tbody>
</table>

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

27E9 POP THE STACK
27EB RETURN WITH POINTER TO BAD INSTRUC.
27EF DIE HORRIBLY
27F2 RETURN

27F3 ********** ERROR RETURN **********

27F3 RETURN WITH POINTER
27F7 EXIT WITH ERROR CODE
27FA RETURN

27FB ********* RELOCATE ABSOLUTE ADDRESS **********

27FB GET PAGE NUMBER TO CHECK
27FD GET NUMBER OF RANGES (LESS ONE) (2800)
2800 IS IT PRIOR TO START OF RANGE? (2881)
2883 YES? >>280C
2805 NO, IS IT AFTER END OF RANGE? (2889)
2888 NO? >>2810
280C ---
280D CHECK EACH RANGE >>2800
280F RETURN

2810 ---
2811 ADD FUDGE FACTOR TO ADDRESS (2891)
2814 AND UPDATE IT
2816 RETURN

2817 ********** BUMP POINTER TO NEXT ADDR **********

2817 ---
2818 ADD LENGTH TO POINTER
281F CHECK TO SEE IF WE ARE DONE
2825 ---
2829 RETURN

ProDOS Relocator -- V1.1.1 -- 18 SEP 84  NEXT OBJECT ADDR: 282A

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

282A ********** COMPUTE INSTRUCTION LENGTH **********

282A A-REG Contains OPCODE
282B ISOLATE LAST TWO BITS FOR LATER
2830 USE LAST 6 BITS AS TABLE INDEX
2832 GET BYTE WITH 4 LENGTHS IN IT (283F)
2835 ---
2836 USING TOP TWO BITS AS INDEX... >>283C
283B SHIFT DOWN THE PROPER LENGTH
283C AND ISOLATE IT IN A-REG
283E RETURN

283F ********** 6502 OP LENGTH TABLE **********

<table>
<thead>
<tr>
<th>EACH BYTE CONTAINS FOUR 2 BIT LENGTHS</th>
</tr>
</thead>
</table>

287F ********** RELOCATION DATA **********

287F RELOCATION CODE (3,2,1)
2880 NUMBER OF RANGES
2881 START OF RANGE PAGES
2889 END OF RANGE PAGES +1
2891 ADDITIVE FACTORS

2899 ********** 2899-28FE NOT USED **********

2899 NOT USED

28FF ********** SET UP RAMDRIVE IN AUXMEM **********

| (THIS ROUTINE PUTS THE RAMDRIVE DEVICE DRIVER IN MEMORY, PUT THE ADDRESS OF THE DRIVER IN THE DEVICE DRIVER ADDRESS LIST, AND ADDS THE RAMDRIVE TO THE ONLINE DEVICE LIST.) |

28FF SUBROUTINE STARTS WITH NOP
2902 RELOCATE RAMDRIVE CALLER NOW AT.. (2C00)
2905 TO HIGH RAM AT.. (FF00)
290D NOW PREPARE TO MOVE
290F RAMDRIVE DEVICE DRIVER
2911 INTO AUX RAM AT $200.
2914 $3C/$3D --> $2A00
2918 $3E/$3F --> $2BFF
291D $42/$43 --> $200
2923 COPY MAIN MEM TO AUX MEM
2924 USE AUXMOVE TO COPY IT <<311>
2929 SLOT 3, DRIVE 2 DEVICE DRIVER.. (BF26)
292C IS AT $FF00
2931 BUMP DEVICE COUNT (BF31)
2937 ADD DEVICE TO ONLINE DEVICE LIST
293C RETURN
<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>293D</td>
<td>293D--29FF NOT USED</td>
</tr>
<tr>
<td>2969</td>
<td></td>
</tr>
<tr>
<td>2A00</td>
<td>RAMDRIVE (/RAM) DEVICE DRIVER</td>
</tr>
<tr>
<td></td>
<td>(COPIED TO AND RUN AT $200 IN AUX RAM)</td>
</tr>
<tr>
<td></td>
<td>(THIS IS THE MAIN PART OF THE DEVICE DRIVER)</td>
</tr>
<tr>
<td></td>
<td>IT IS CALLED BY THE RAMDRIVE CALLER</td>
</tr>
<tr>
<td></td>
<td>WHICH IS LOCATED AT $FF00 IN MAIN MEMORY.)</td>
</tr>
<tr>
<td>2A00</td>
<td>SAVE THE $0STORE SETTING (C018)</td>
</tr>
<tr>
<td>2A04</td>
<td>FORCE RAM READ/WRITE (C000)</td>
</tr>
<tr>
<td>2A09</td>
<td>COPY INPUT PARAMETERS</td>
</tr>
<tr>
<td>2A0B</td>
<td>TO AUX PAGE 3 (03BD)</td>
</tr>
<tr>
<td>2A11</td>
<td>FIRST TIME IN OR FORMAT COMMAND? (03BC)</td>
</tr>
<tr>
<td>2A14</td>
<td>NO, SKIP FORMAT LOGIC &gt;&gt;2A4F</td>
</tr>
<tr>
<td></td>
<td>********** FORMAT RAMDRIVE **********</td>
</tr>
<tr>
<td>2A16</td>
<td>YES, SAVE BLOCK WANTED</td>
</tr>
<tr>
<td>2A18</td>
<td>PAGES SE AND $F ARE ACTUAL DIRECTORY</td>
</tr>
<tr>
<td>2A1A</td>
<td>ZERO THE DIRECTORY BLOCK (0333)</td>
</tr>
<tr>
<td>2A1F</td>
<td>COPY VOLUME NAME (SF3,&quot;RAM&quot;) (03D2)</td>
</tr>
<tr>
<td>2A22</td>
<td>TO VOLUME DIRECTORY BLOCK (0204)</td>
</tr>
<tr>
<td>2A28</td>
<td>LAST BYTE IN VOLUME BITMAP</td>
</tr>
<tr>
<td>2A2A</td>
<td>IS AN $FE (03D1)</td>
</tr>
<tr>
<td>2A2D</td>
<td>$FF TO ACCUM.</td>
</tr>
<tr>
<td>2A30</td>
<td>14 $FF'S TO BITMAP (03C2)</td>
</tr>
<tr>
<td>2A36</td>
<td>SET FIRST BITMAP BYTE TO ZERO (03C2)</td>
</tr>
<tr>
<td>2A39</td>
<td>COPY 8 BYTES</td>
</tr>
<tr>
<td>2A3B</td>
<td>OF DIRECTORY DATA (03D6)</td>
</tr>
<tr>
<td>2A3E</td>
<td>VOLUME DIRECTORY BLOCK (02E2)</td>
</tr>
<tr>
<td>2A44</td>
<td>WAS THIS A FORMAT COMMAND? (03BC)</td>
</tr>
<tr>
<td>2A47</td>
<td>YES, DONE. &gt;&gt;2AAA</td>
</tr>
<tr>
<td>2A49</td>
<td>NO, SET FLAG &amp; CONTINUE WITH READ/ WRITE (03BC)</td>
</tr>
<tr>
<td>2A4C</td>
<td>RESTORE BLOCK NUMBER (03C1)</td>
</tr>
<tr>
<td></td>
<td>********** READ/ WRITE RAMDRIVE BLOCK **********</td>
</tr>
<tr>
<td>2A4F</td>
<td>CONVERT BLOCK NUMBER TO PAGE NUMBER (03C1)</td>
</tr>
<tr>
<td>2A55</td>
<td>THIS PAGE IN HIGH RAM?</td>
</tr>
<tr>
<td>2A57</td>
<td>YES &gt;&gt;2A63</td>
</tr>
<tr>
<td>2A59</td>
<td>NO, IS IT BLOCK 37 (VOLUME BIT MAP)</td>
</tr>
<tr>
<td>2A5B</td>
<td>NO &gt;&gt;2A60</td>
</tr>
<tr>
<td>2A5D</td>
<td>YES, DUMMY UP A PHONY BITMAP BLOCK &gt;&gt;03BC</td>
</tr>
<tr>
<td>2A60</td>
<td>ELSE, NORMAL READ/ WRITE &gt;&gt;0342</td>
</tr>
<tr>
<td>2A63</td>
<td>SAVE PAGE NUMBER</td>
</tr>
<tr>
<td>2A64</td>
<td>FIND IT IN MEMORY &lt;02E5&gt;</td>
</tr>
<tr>
<td>2A67</td>
<td>REMEMBER READ/ WRITE STATUS</td>
</tr>
<tr>
<td>2A69</td>
<td>WRITING? &gt;&gt;2A88</td>
</tr>
<tr>
<td>2A6A</td>
<td>GET SAVED PAGE NUMBER</td>
</tr>
<tr>
<td>2A6B</td>
<td>DOES OPERATION INVOLVE BANK1?</td>
</tr>
<tr>
<td>2A6D</td>
<td>NO, USE BANK2 &gt;&gt;2A73</td>
</tr>
<tr>
<td>2A6F</td>
<td>YES, FORCE IT TO $DXX</td>
</tr>
<tr>
<td>2A71</td>
<td>AND USE BANK1 OF AUX HIGH RAM &gt;&gt;2A79</td>
</tr>
<tr>
<td>2A73</td>
<td>USE BANK2 OF AUX HIGH RAM (C083)</td>
</tr>
<tr>
<td>2A76</td>
<td>AND WRITE ENABLE IT (C083)</td>
</tr>
<tr>
<td>2A79</td>
<td>SAVE PAGE NUMBER IN BLOCK (03C1)</td>
</tr>
<tr>
<td>2A7C</td>
<td>PRESERVE HIS BUFFER ADDR (03C0)</td>
</tr>
<tr>
<td>2A80</td>
<td>DURING THE FOLLOWING TRANSFER... (03BF)</td>
</tr>
<tr>
<td>2A83</td>
<td>SELECT AUX HIGH RAM (C009)</td>
</tr>
<tr>
<td>2A88</td>
<td>USE RAMDRIVE BUFFER AS AN &quot;IN BETWEEN&quot; (03C0)</td>
</tr>
<tr>
<td>2A8B</td>
<td>AREA WHEN TRANSFERING TO/FROM AUX HIGH RAM.</td>
</tr>
<tr>
<td>2A9D</td>
<td>PRETEND THAT WAS CALLER'S BUFFER (03BF)</td>
</tr>
<tr>
<td>2A99</td>
<td>AND SET UP POINTERS AGAIN &lt;02E5&gt;</td>
</tr>
<tr>
<td>2A94</td>
<td>COPY BLOCK TO OR FROM RAMDRIVE BUFFER</td>
</tr>
<tr>
<td>2A98</td>
<td>THEN BACK TO MAIN ZERO PAGE (C000)</td>
</tr>
<tr>
<td>2A92</td>
<td>RESTORE CALLER'S BUFFER ADDRESS (03BF)</td>
</tr>
<tr>
<td>2A99</td>
<td>READING OR WRITING?</td>
</tr>
<tr>
<td>2AA</td>
<td>IF WRITING, DONE &gt;&gt;2A85</td>
</tr>
<tr>
<td>2AAC</td>
<td>IF READING, WRITE ENABLE HIGH RAM (BANK1) (C088)</td>
</tr>
<tr>
<td>2A02</td>
<td>AND COPY RAMDRIVE BUFFER TO HIS BUFFER &lt;02BE</td>
</tr>
<tr>
<td>2AB5</td>
<td>THEN EXIT &gt;&gt;03DE</td>
</tr>
<tr>
<td>2A88</td>
<td>IF WRITING, COPY HIS BLOCK TO RAMDRIVE BUFFER &lt;02BE</td>
</tr>
<tr>
<td>2AB9</td>
<td>THEN COPY RAMDRIVE BUFFER TO AUX HIGH RAM &gt;&gt;026A</td>
</tr>
<tr>
<td>2ABE</td>
<td>COPY BLOCK IN MAIN 48K ***********************</td>
</tr>
<tr>
<td>2ABE</td>
<td>THIS ENTRY IS FOR THE RAMDRIVE BUFFER</td>
</tr>
<tr>
<td>2AC0</td>
<td>THIS ENTRY ASSUMES AUX MEM PAGE NUMBER IN ACCUM (03C1)</td>
</tr>
<tr>
<td>2AC3</td>
<td>THIS ENTRY ASSUMES PAGE NUMBER ALREADY SET &lt;02E5&gt;</td>
</tr>
<tr>
<td>2AC6</td>
<td>WRITING TO RAMDISK? &gt;&gt;2ADB</td>
</tr>
<tr>
<td>2AC8</td>
<td>NO, WRITE TO MAIN 48K RAM (C004)</td>
</tr>
<tr>
<td>2ACC</td>
<td>COPY BLOCK AUX MEM --&gt; MAIN MEM</td>
</tr>
<tr>
<td>2ADB</td>
<td>WRITE TO AUX MEM AGAIN (C005)</td>
</tr>
<tr>
<td>2AD0</td>
<td>DONE (RETURN HERE AFTER FOLLOWING JUMP)</td>
</tr>
<tr>
<td>2ADD</td>
<td>---</td>
</tr>
<tr>
<td>2ADD</td>
<td>GO BACK TO MAIN MEM PART OF DRIVER (03ED)</td>
</tr>
<tr>
<td>2AE0</td>
<td>TO COPY MAIN MEM --&gt; AUX MEM</td>
</tr>
</tbody>
</table>
2AE5 ******************** SET BUFFER AND BLOCK ADDRESSES ********************

2AE5 GET COMMAND (03BD)
2AE8 READ OR WRITE?
2EA9 WRITE? >>2B08
2EBA NO, GET HIGH BYTE OF BUFFER TO BE READ (03Cu)
2EBAF AND LOW BYTE OF BUFF ADDRESS (03BF)
2EAF $42/43 --> FIRST PAGE OF BUFFER
2EA7 $40/41 --> SECOND PAGE OF BUFFER
2EAF GET PAGE NUMBER (03C1)
2EAE $3C/3D --> BLOCK IN RAMDRIVE
2EBAF $3E/3F --> SECOND PAGE OF SAME
2EB0 ALWAYS BRANCH AROUND WRITE CODE >>2B23

2B00 WRITE, (03C0)
2B04F $3C/3D --> MAIN MEMORY ADDRESS OF BUFFER TO BE WRITTEN (03BF)
2B02 $3E/3F --> SECOND PAGE OF SAME
2B01F $42/43 --> BLOCK IN RAMDRIVE
2B01F $40/41 --> SECOND PAGE OF SAME
2B02F SET SECOND PAGE ADDRESSES
2B07F EXIT

2B28 ****************** SEND HIM A DUMMY BLOCK OF ZEROES********************

2B28F ZERO RAMDRIVE BUFFER IN CASE READING <0331>
2B2B COPY BETWEEN RAMDRIVE BUFFER AND HIS BUFFER <02C3>
2B2BF EXIT >>03DE

2B31 ****************** ZERO BLOCK BUFFER ******************

2B31F ZERO RAMDRIVE BUFFER
2B33F ZERO BLOCK INDICATED BY ACCUM. (03C1)
2B35F SET UP BUFFER POINTERS <02E5>
2B3AF ZERO BOTH PAGES OF BLOCK
2B41F AND EXIT

2B42 **************** READ/WRITE IN LOW 48K ******************

2B42F BLOCK 2 (VOLUME DIRECTORY)?
2B44 NO >>2B4A
2B46 YES, CONVERT IT BLOCK 7
2B4Af AND GO TO I/O NOW >>2B5B
2B4AF ELSE, LESS THAN BLOCK 87 (BUG--$D SHOULD BE $F111)
2B4F YES, RETURN WITH DUMMY ZERO BLOCK. >>2B28
2B45F START MSB AT ZERO
2B50F GET ORIGINAL BLOCK NUMBER
2B52F BLOCK $50 THROUGH $5F?
2B54F NO >>2B5B
2B56F YES, ADJUST TO $D THROUGH $F
2B58F AND USE $1A00 $1FFF IN RAMDRIVE. >>0385

2BE5 ELSE, FOR BLOCKS $8 THRU $5C
2BE5 SUBTRACT 8
2BE5F AND DIVIDE BY 17 ($11)
2BE6F XREG IS QUOTIENT
2BEBF HAS TO BRANCH >>2B5E
2BF6F AND XREG IS REMAINDER
2BF9F REMAINDER OF 17
2BF6F NO >>2B73
2BF6F YES, EVERY 17TH BLOCK GOES
2BF7 IN $1000-$1BFF AREA
2BF7F BY ADDING 8 TO QUOTIENT
2BF7F AND GO DO IT >>2B5F
2BF7F BUMP QUOTIENT (START AT $2FFF)
2BF7F SHIFT IT TO TOP NIBBLE OF BYTE
2BF7F GOT A REMAINDER? >>2B01
2BF7F IF SO, DECREMENT IT (NOT USING 1)
2BF7F THEN ADD INTO TOP NIBBLE
2BF7F TO FORM $1B THRU $5F (03C1)
2BF5F BLOCK*2 FOR PAGE NUMBER
2BF6F COPY THE BLOCK <02C0>
2BF9F THEN EXIT >>03DE

2B8C *************** READ/WRITE BITMAP BLOCK ******************

2B8C COPY USE RAMDRIVE BUFFER (NO ACTUAL BITMAP BLOCK)
2B91F SET UP BUFFER POINTERS <02E5>
2B94F WRITING? >>2B89
2B96F NO, READING - ZERO THE RAMDRIVE BUFFER <0336>
2B98F COPY BITMAP IMAGE TO RAMDRIVE BUFFER (03C2)
2B95F COPY BLOCK BACK TO CALLER'S BUFFER <02C3>
2B98F THEN EXIT >>03DE

2B99F WRITING, COPY HIS BUFFER TO RAMDRIVE BUFFER <02C3>
2BACF SET UP BUFFER POINTERS <02E5>
2BBDF COPY 16 BITMAP BYTES FROM RAMDRIVE BUFFER
2BB6F INTO PAGE 3 BITMAP IMAGE (03C2)
2BB9F THEN EXIT >>03DE

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

2BBD FIRST TIME ENTRY FLAG
2BBDF COMMAND FROM PARM LIST
2BBEF UNIT NUMBER FROM PARM LIST
2BFFF BUFFER ADDRESS FROM PARM LIST
2BC1F BLOCK NUMBER FROM PARM LIST
<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>2BC2</td>
<td>BIT MAP IMAGE FOR RAM DRIVE</td>
</tr>
<tr>
<td>2B53</td>
<td>'RAM'</td>
</tr>
<tr>
<td>2B56</td>
<td>ACCESS, ENTRY LENGTH</td>
</tr>
<tr>
<td>2B5F</td>
<td>NUMBER OF ENTRIES</td>
</tr>
<tr>
<td>2B6A</td>
<td>FILE COUNT</td>
</tr>
<tr>
<td>2B6B</td>
<td>BIT MAP BLOCK POINTER</td>
</tr>
<tr>
<td>2B6D</td>
<td>BLOCKS ON DISK</td>
</tr>
</tbody>
</table>

**2BDE********** EXIT TO MAIN MEMORY******************************

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>2BDE</td>
<td>WRITE ENABLE HIGH RAM (BANK1) (C88B)</td>
</tr>
<tr>
<td>2B5E</td>
<td>RESTORE $STORE STATUS &gt;&gt;2BEA</td>
</tr>
<tr>
<td>2B57</td>
<td>$STORE WAS ON (C801)</td>
</tr>
<tr>
<td>2B6A</td>
<td>GO AROUND MEMORY USED BY XFER &gt;&gt;03EF</td>
</tr>
<tr>
<td>2B6D</td>
<td>LOW-ORDER BYTE AND</td>
</tr>
<tr>
<td>2B0E</td>
<td>HIGH-ORDER BYTE USED BY XFER ROUTINE</td>
</tr>
<tr>
<td>2BEF</td>
<td>RETURN TO $FF64 (NORMAL EXIT)</td>
</tr>
<tr>
<td>2BFB</td>
<td>USE ROM XFER ROUTINE TO DO IT &gt;&gt;C314</td>
</tr>
<tr>
<td>2BFE</td>
<td>TWO BYTES NOT USED</td>
</tr>
</tbody>
</table>

**2C00********** RAMDRIVE CALLER (RUNS AT $FF80)******************************

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>2C00</td>
<td>(USED TO CALL MAIN PART OF RAMDRIVE DEVICE DRIVER WHICH IS AT $2888 IN RAM.)</td>
</tr>
<tr>
<td>2C03</td>
<td>ROUTINE AT $FF62 IS USED TO TRANSFER DATA FROM MAIN TO AUX MEM.)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>2C03</td>
<td>SAVE 2PAGE STUFF I WILL Clobber</td>
</tr>
<tr>
<td>2C5D</td>
<td>FROM $3C THRU $47 (FF81)</td>
</tr>
<tr>
<td>2C5D</td>
<td>SAVE $3ED/E THAT XFER ROUTINE WILL Clobber ($3ED)</td>
</tr>
<tr>
<td>2C16</td>
<td>COMMAND = STATUS?</td>
</tr>
<tr>
<td>2C18</td>
<td>IF SO, SIMPLE EXIT WILL DO &gt;&gt;2C44</td>
</tr>
<tr>
<td>2CA1</td>
<td>ELSE, TOO BIG A COMMAND NUM?</td>
</tr>
<tr>
<td>2C1C</td>
<td>IF SO, ERROR &gt;&gt;2C3B</td>
</tr>
<tr>
<td>2C1E</td>
<td>ELSE, INVERT BITS OF CMD</td>
</tr>
<tr>
<td>2C20</td>
<td>AND SAVE IT</td>
</tr>
<tr>
<td>2C22</td>
<td>FORMAT7 &gt;&gt;2C2C</td>
</tr>
<tr>
<td>2C24</td>
<td>NO, CHECK BLOCK NUMBER</td>
</tr>
<tr>
<td>2C25</td>
<td>MUST BE &lt;128 FOR RAMDRIVE</td>
</tr>
<tr>
<td>2C2C</td>
<td>GOING TO $280 IN AUX MEMORY</td>
</tr>
</tbody>
</table>

**FF33**

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>2C3B</td>
<td>USE XFER ROUTINE TO GET THERE &gt;&gt;C314</td>
</tr>
<tr>
<td>2C3D</td>
<td>I/O ERROR RETURN CODE</td>
</tr>
<tr>
<td>2C64</td>
<td>EXIT &gt;&gt;2C41</td>
</tr>
<tr>
<td>2CE4</td>
<td>WRITE PROTECTED RETURN CODE</td>
</tr>
<tr>
<td>2C42</td>
<td>ERROR EXIT &gt;&gt;2C47</td>
</tr>
</tbody>
</table>
**ProDOS MLI -- V1.1.1 -- 18 SEP 84**

**ADDR** | **DESCRIPTION/CONTENTS**
--- | ---
D700 | **MODULE STARTING ADDRESS**

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRODOS MACHINE LANGUAGE INTERFACE</td>
</tr>
<tr>
<td>THIS CODE IS MOVED INTO HIGH</td>
</tr>
<tr>
<td>RAM ($DE00-$FEFF) BY THE</td>
</tr>
<tr>
<td>PRODOS RELOCATOR,</td>
</tr>
<tr>
<td>IF PERFORMS ALL FILE MANAGEMENT</td>
</tr>
<tr>
<td>AND OTHER SYSTEM FUNCTIONS AND</td>
</tr>
<tr>
<td>SUPPORTS THE HARDWARE IN A</td>
</tr>
<tr>
<td>DEVICE INDEPENDENT WAY.</td>
</tr>
<tr>
<td>VERSION 1.1.1 -- 18 SEP 84</td>
</tr>
</tbody>
</table>

**D700 ******* ZERO PAGE USAGE *******

<table>
<thead>
<tr>
<th>ADDR</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0040</td>
<td>Pointer to callers parmlist</td>
</tr>
<tr>
<td>0041</td>
<td>-- device driver parmlist --</td>
</tr>
<tr>
<td>0042</td>
<td>Command</td>
</tr>
<tr>
<td>0043</td>
<td>Unit Number</td>
</tr>
<tr>
<td>0044</td>
<td>Buffer Pointer</td>
</tr>
<tr>
<td>0045</td>
<td></td>
</tr>
<tr>
<td>0046</td>
<td>Block Number</td>
</tr>
<tr>
<td>0047</td>
<td>I/O Pointer - Index Block or...</td>
</tr>
<tr>
<td>0048</td>
<td>pointer into $F600 work buffer or...</td>
</tr>
<tr>
<td>0049</td>
<td>caller's pathname buffer pointer</td>
</tr>
<tr>
<td>004A</td>
<td>I/O Pointer - Data Block</td>
</tr>
<tr>
<td>004B</td>
<td>I/O Pointer - Data Block</td>
</tr>
<tr>
<td>004C</td>
<td>I/O Pointer - Data Block</td>
</tr>
<tr>
<td>004E</td>
<td>I/O Pointer - Caller's Data or...</td>
</tr>
<tr>
<td>004E</td>
<td>buffer pointer passed in parmlist or...</td>
</tr>
<tr>
<td>004E</td>
<td>old I/O buffer</td>
</tr>
</tbody>
</table>

**D700 ******* MLI ERROR CODES *******

<table>
<thead>
<tr>
<th>ADDR</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>No Error</td>
</tr>
<tr>
<td>0001</td>
<td>Bad call type</td>
</tr>
<tr>
<td>0004</td>
<td>Bad parameter count</td>
</tr>
<tr>
<td>0025</td>
<td>Interrupt Table full</td>
</tr>
<tr>
<td>0027</td>
<td>I/O Error</td>
</tr>
<tr>
<td>0028</td>
<td>No device connected</td>
</tr>
<tr>
<td>002B</td>
<td>Write protected</td>
</tr>
</tbody>
</table>

**D700 ******* SCREEN LOCATIONS *******

<table>
<thead>
<tr>
<th>ADDR</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0750</td>
<td>For direct movement of text to screen</td>
</tr>
<tr>
<td>07D0</td>
<td></td>
</tr>
<tr>
<td>07F1</td>
<td></td>
</tr>
<tr>
<td>07F2</td>
<td></td>
</tr>
<tr>
<td>07F3</td>
<td></td>
</tr>
<tr>
<td>07F4</td>
<td></td>
</tr>
<tr>
<td>07F5</td>
<td></td>
</tr>
<tr>
<td>07F6</td>
<td></td>
</tr>
<tr>
<td>07F7</td>
<td></td>
</tr>
<tr>
<td>07F8</td>
<td>Slot in use</td>
</tr>
</tbody>
</table>

**D700 ******* SYSTEM GLOBAL PAGE EQUATES *******

<table>
<thead>
<tr>
<th>ADDR</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BF00</td>
<td>Jump to MLI entry point</td>
</tr>
<tr>
<td>BF03</td>
<td>JSFASRE (Jump to $ECF, QUIT code)</td>
</tr>
<tr>
<td>BF06</td>
<td>DATETIME vector</td>
</tr>
<tr>
<td>BF09</td>
<td>Jump to System Error</td>
</tr>
<tr>
<td>BF0C</td>
<td>Jump to System Death Handler</td>
</tr>
<tr>
<td>BF0F</td>
<td>System Error number</td>
</tr>
<tr>
<td>BF10</td>
<td>Device Driver address table</td>
</tr>
<tr>
<td>BF30</td>
<td>Slot/Drive last device</td>
</tr>
<tr>
<td>BF31</td>
<td>Count (-1) active devices</td>
</tr>
<tr>
<td>BF32</td>
<td>List of active devices by DEVID</td>
</tr>
<tr>
<td>BF58</td>
<td>Memory BITMAP for low 40K</td>
</tr>
<tr>
<td>BF70</td>
<td>Open file 1 buffer address</td>
</tr>
</tbody>
</table>
ProDOS MLI -- V1.1.1 -- 18 SEP 84

**ADDRESSES AND DESCRIPTIONS**

- BF7E Open file buffer address
- BF80 Interrupt handler 1
- BF82 Interrupt handler 2
- BF84 Interrupt handler 3
- BF86 Interrupt handler 4
- BF88 A reg save during interrupt
- BF89 X reg save during interrupt
- BF8A Y reg save during interrupt
- BF8B S reg save during interrupt
- BF8C P reg save during interrupt
- BF8E Interrupt return address
- BF90 Date/Time
- BF94 File open LEVEL
- BF95 Backup bit (0 = no prefix)
- BF9A MLI active flag
- BF9C Last MLI call return address
- BF9E MLI X reg save area
- BF9F MLI Y reg save area
- BFAC HIGH RAM entry/exit routines
- BFD0 Interrupt entry/exit routines
- BF4 Bank switch saved state ($E000 byte)

---

**SOFT SWITCHES**

- C05C Reset 80 column mode
- C051 Set TEXT mode
- C053 Set Mixed text/graphics
- C054 Display Primary page
- C055 Set LOGES graphics mode
- C056 Set LOGES graphics mode
- C05F Set alternate I/O ROMs

---

**PATHNAME - DATA AREA**

- | L1 | NAME1 | L2 | NAME2 | ... | 00

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

---

**FILE CONTROL BLOCKS**

- D800 path name buffer

---

**VOLUME CONTROL BLOCKS**

- VCB0 starts here...
- D900 Length (00001LLL)
- D901 File Name (Max 15)
- D910 Unit Number
- D911 Files Open Flag (if $FF)
- D912 Total Blocks
- D914 Blocks Free
- D916 Block Number of Vol Dir Key Block
- D918 not used
- D919 not used
- D91A Bit Map Pointer
- D91E Count of open files

---

**FILE ID**

- D901 Device Number
- D902 Dir Block HDR for Dir describing this File
- D904 Dir Block containing entry itself
- D906 File entry # in this Directory

---

**Storage Type Flags**

- IXXX XXXX Index Block Buffer Changed
- X1XX XXXX Data Block Buffer Changed
- X1XX XX1X Unused
- X1XX XXXX Directory entry needs update
- XXXX IXXX Storage Type Changed
- XXXX XXXX Allocate new Master Index Block
- XXXX XX1X Allocate new Sub Index Block

---

**Allocate new Data Block**

- D809 Access Byte
- D80A Newline Character
- D80B Buffer Number (REP NUM + 2)
- D80C Master Index/Key Block Number
- D80E Current Index Block
- D810 Current Data Block
- D812 Mark
- D815 End of File
- D818 Blocks Used
- D81A not used
- D81B Level
- D81C Flag - Write occurred if MSB on
- D81D not used
- D81E Newline Enable Mask

---

**FCB0 through FCB7**
**Beneath Apple ProDOS Supplement**

ProDOS MLI -- V1.1.1 -- 18 SEP 84

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>D920</td>
<td>VCB1 through VCB7</td>
</tr>
<tr>
<td>DA00</td>
<td>*********** BITMAP BUFFER *************************</td>
</tr>
<tr>
<td>DA00</td>
<td>Buffer 1st half</td>
</tr>
<tr>
<td>DB00</td>
<td>Buffer 2nd half</td>
</tr>
<tr>
<td>DCC0</td>
<td>********** PRIMARY BUFFER*****************************</td>
</tr>
<tr>
<td>DDA0</td>
<td>(Used for several things. VOL DIR HDR is mapped into it below)</td>
</tr>
<tr>
<td>DCC0</td>
<td>Pointer Fields</td>
</tr>
<tr>
<td>DCC4</td>
<td>Type/Length (TTLTLLLL)</td>
</tr>
<tr>
<td>DCC5</td>
<td>Volume Name (Max 15)</td>
</tr>
<tr>
<td>DCC4</td>
<td>Reserved</td>
</tr>
<tr>
<td>DCC4</td>
<td>Creation Datetime</td>
</tr>
<tr>
<td>DCCD</td>
<td>Version</td>
</tr>
<tr>
<td>DCD2</td>
<td>Min Version</td>
</tr>
<tr>
<td>DCD2</td>
<td>Access Byte</td>
</tr>
<tr>
<td>DCD3</td>
<td>Entry Length</td>
</tr>
<tr>
<td>DCD4</td>
<td>Entries per Block</td>
</tr>
<tr>
<td>DCD5</td>
<td>File Count</td>
</tr>
<tr>
<td>DCD7</td>
<td>Bitmap Pointer</td>
</tr>
<tr>
<td>DCD9</td>
<td>Total Blocks</td>
</tr>
<tr>
<td>DC02</td>
<td>(remainder of first page of block)</td>
</tr>
<tr>
<td>DD00</td>
<td>(second page of block)</td>
</tr>
<tr>
<td>DE00</td>
<td>********** MLI MAIN ENTRY POINT ******************</td>
</tr>
</tbody>
</table>

**DE00**

Clear decimal mode

Save Registers (BF9F)

Set ($40) -> Address of function code -1

Set CMADDR -> True return address

Init Global Page System error to 0 (BF0F)

Get Function Code

Build hash index into Command Table (X reg)

Is this code valid?

No >> XEA?

Set ($40) -> Parameter list

Get parameter count required (FD65)

None? >> DE60

No - is parameter count correct?

No >> DEAB

Check class of function (FD65)

Quit?

yes >> DE50

no,

$BX - Calls to I/O Drivers >> DE66

$CX/DX - Non System calls >> DE71

---

**DE54**

Else, %4X - Interrupt support

Isolate type (DEALLOC = 1, ALLOC = 0)

Call Interrupt Support <DEF3>

Then Exit to Caller >>DE78

Go to quit code via global page >>BF03

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

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

Call Date/Time driver <BF06>

and exit to caller >>DE78

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

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

$8d - Read Block

$81 - Write Block

---

Set %42 -> 1 for READ, 2 for WRITE

Do Block I/O <DEB2>

Then Exit to Caller >>DE78

****** MLI ISO MTFN CALLS ******

Category function Index

Perform function and exit to caller <E047>

---

Clear Backup

Error occurred?

Save test results

Disable interrupts

MLI no longer active (BF9B)

Get test results back

Store in X reg

Set up Return Address on stack (BF9D)

Put test results on stack

Put error code in A reg

Restore X reg (BF9E)

Restore Y reg (BF9F)

Put error code on stack

Get RAM/RPM orientation (BF4)

Exit via RAM Global Page >>BFA8
DEA2 ********** NO DEVICE CONNECTED **************************************

DEA2 ---
DEA4 Call System Error Handler (Global Page) <BF09>

DEA7 ********** BAD SYSTEM CALL NUMBER **************************************

DEA7 ---
DEA9 Branch always taken >>DEAD

DEA8 ********** BAD PARAMETER COUNT **************************************

DEA8 ---
DEAD Call System Error Handler <DED7>
DEB0 Exit to Caller >>DE78

DEB2 ********** BLOCK I/O SETUP **************************************

DEB2 ---
DEB4 Save Old Processor Flags
DEB5 Disable Interrupts
DEB6 Copy Parameters to $43-$47
DEB8 Save Starting Buffer Page in $4F
DEC3 Find last page + 1
DEC6 Round up if Buffer not page aligned >>DEC9
DEC9 Is this Memory already in use? <FC9F>
DEC2C Yes, then exit with error >>DED6
DEC3E No, do Block I/O <DEDA>
DED1 Error? >>DED6
DED3 No, then exit normally
DED5 RETURN
DED6 Error Exit
DED7 Call System Error Handler <BF09>

DEDA ********** Block I/O *********************************************************

DEDA ---
DEDC Force off unused UNIT bits
DEDE Put Drive number in X reg
DEEF Put Device Handler Address in Jump Vector (FEF5)
DEFF Exit through Device Handler >>FEF5

DEF3 ********** Interrupt Handler *********************************************************

ALLOC/DEALLOC

DEF3 Save Call Type
DEF5 Which Type?
DEF6 DEALLOC? >>DF24

ALLOC

DEF8 ---
DEFA Look for empty slot (BF7E)
DEF1 His Address better be non-zero
DEF5 Store Address of His routine in Global Page (BF7E)
DEF8E And return the position number we used
DEF4 Exit
DEF5 Skip this Vector
DEFE Last one?
DEF9 No, check another >>DEFA
DF1B Yes, Table Full Error
DF1D Always taken >>DF21
DF1F Bad Parameter Error
DF21 Call System Error Handler <BF09>

DEALLOC

DF2B ---
DF26 Get Position Number
DF28 Can’t be zero >>DF1F
DF2C Or greater than 4 >>DF1F
DF2F Starting Buffer from it
DF32 And zero His Vector (BF7E)
DF39 Then Exit

DF3A ********** IMQ Handler *********************************************************

DF3A ---
DF3C Save A reg from Monitor (BF88)
DF3F And X,Y,S and P (BF89)
DF49 Is this ROM enhanced? (DFD8)
DF4C Yes, skip three pulls >>DF5A
DF53 And RTI Address (BF8E)
DF5A Replace stack to original condition
DF5E Save active slot index (DFCE)
DF61 In bottom half of stack?
DF64 Yes, pop off 16 bytes and save them
DF66 ---
DF6D Save $FA - $FF (top of zero page)
DF6F ---
DF77 Is there a User Vector #1 (BF81)
DF7A No >>DF81
DF7C Yes, call it <DFD9>
DF7F His interrupt? >>DFA4
DF81 Is there a User Vector #2 (BF83)
DF84 No >>DF8B
DF86 Yes, call it <DFDC>
DF89 His interrupt? >>DFA4
DF8B Is there a User Vector #3 (BF35)
<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFBE</td>
<td>No &gt;&gt;DF95</td>
</tr>
<tr>
<td>DF98</td>
<td>Yes, call it &lt;DFDF&gt;</td>
</tr>
<tr>
<td>DFEA</td>
<td>His interrupt? &gt;&gt;DF9A</td>
</tr>
<tr>
<td>DF95</td>
<td>Is there a User Vector #4 (BF87)</td>
</tr>
<tr>
<td>DF98</td>
<td>No &gt;&gt;DF9F</td>
</tr>
<tr>
<td>DF9A</td>
<td>Yes, call it &lt;DF2E&gt;</td>
</tr>
<tr>
<td>DF9D</td>
<td>His interrupt? &gt;&gt;DF9A</td>
</tr>
<tr>
<td>DF9F</td>
<td>Indicate error type 1</td>
</tr>
<tr>
<td>DF9A</td>
<td>Call System Death Handler &lt;BF0C&gt;</td>
</tr>
<tr>
<td>DF96</td>
<td>Interrupt Serviced</td>
</tr>
<tr>
<td>DFEA</td>
<td>Restroom zero page (FDF5)</td>
</tr>
<tr>
<td>DF96</td>
<td>And stack (BF6B)</td>
</tr>
<tr>
<td>DFBE</td>
<td>Is this enhanced ROM? (DFD8)</td>
</tr>
<tr>
<td>DF91</td>
<td>Yes, skip some stuff we used to have to do &gt;&gt;DFDF</td>
</tr>
<tr>
<td>DF9C</td>
<td>Reload X and Y (BF9A)</td>
</tr>
<tr>
<td>DF9A</td>
<td>Disable I/O ROMs (CFFP)</td>
</tr>
<tr>
<td>DF9D</td>
<td>Replace active slot number (C100)</td>
</tr>
<tr>
<td>DFD5</td>
<td>Exit from interrupt &gt;&gt;BF0D</td>
</tr>
<tr>
<td>DF98</td>
<td>ENHANCE FLAG. Set to 1 by RELOCATOR if new type ROM found. (That is, if ROM is G vector jumps below $0000)</td>
</tr>
<tr>
<td>DFD9</td>
<td>User Interrupt Handlers (#1 - #4) &gt;&gt;BF08</td>
</tr>
<tr>
<td>DFE5</td>
<td>Save Error Code (BF0F)</td>
</tr>
<tr>
<td>DFEF</td>
<td>Pop out of subroutine</td>
</tr>
<tr>
<td>DFFE</td>
<td>Exit to caller with Error Code (BF0F)</td>
</tr>
<tr>
<td>DFFF</td>
<td>RETURN</td>
</tr>
</tbody>
</table>

**SYSTEM DEATH HANDLER**

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFF1</td>
<td>Entry from System Global Page here</td>
</tr>
<tr>
<td>DFF2</td>
<td>Turn off 80 column card (C06C)</td>
</tr>
<tr>
<td>DFF3</td>
<td>Select standard Text display (C051)</td>
</tr>
<tr>
<td>E001</td>
<td>Blank out a line</td>
</tr>
<tr>
<td>E003</td>
<td>---</td>
</tr>
<tr>
<td>E006</td>
<td>Print &quot;INSERT SYSTEM DISK AND RESTART&quot; (FE1E)</td>
</tr>
<tr>
<td>E012</td>
<td>Go into infinite loop if no error code &gt;&gt;E044</td>
</tr>
<tr>
<td>E016</td>
<td>&quot;=&quot; (E07F)</td>
</tr>
<tr>
<td>E01B</td>
<td>&quot;E&quot; (E07F2)</td>
</tr>
<tr>
<td>E020</td>
<td>&quot;R&quot; (E07F3)</td>
</tr>
<tr>
<td>E021</td>
<td>&quot;M&quot; (E07F4)</td>
</tr>
<tr>
<td>E027</td>
<td>Convert error code to Hex</td>
</tr>
<tr>
<td>E033</td>
<td>And print it (E07F6)</td>
</tr>
<tr>
<td>E037</td>
<td>Second digit also</td>
</tr>
<tr>
<td>E044</td>
<td>Infinite loop &gt;&gt;E044</td>
</tr>
</tbody>
</table>

**PERFORM FILING OR HOUSEKEEPING FUNCTIONS**

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>E047</td>
<td>Save function index (FEB7)</td>
</tr>
<tr>
<td>E04A</td>
<td>Get INFO flags for this command (FDCC)</td>
</tr>
<tr>
<td>E04D</td>
<td>Times</td>
</tr>
<tr>
<td>E04E</td>
<td>Store Command Number times 2 (FEB3)</td>
</tr>
<tr>
<td>E053</td>
<td>And use it to index into Address Table</td>
</tr>
<tr>
<td>E057</td>
<td>Set up Jump Vector with this function's (FEB5)</td>
</tr>
<tr>
<td>E05A</td>
<td>.handler address (FAD6)</td>
</tr>
<tr>
<td>E060</td>
<td>Signal backup required after call</td>
</tr>
<tr>
<td>E065</td>
<td>PATHNAME not required? &gt;&gt;E06C</td>
</tr>
<tr>
<td>E067</td>
<td>Required - parse and validity check &lt;E06A&gt;</td>
</tr>
<tr>
<td>E06A</td>
<td>Bad Name? &gt;&gt;E063</td>
</tr>
<tr>
<td>E06C</td>
<td>Reference Number in list? (FEB3)</td>
</tr>
<tr>
<td>E06F</td>
<td>No &gt;&gt;E076</td>
</tr>
<tr>
<td>E071</td>
<td>Yes - check it out &lt;E1D0&gt;</td>
</tr>
<tr>
<td>E074</td>
<td>Bad Number? &gt;&gt;E083</td>
</tr>
<tr>
<td>E076</td>
<td>Date/Time in list? (FEB3)</td>
</tr>
<tr>
<td>E079</td>
<td>No &gt;&gt;E07E</td>
</tr>
<tr>
<td>E07B</td>
<td>Yes - set System date just in case &lt;BF08&gt;</td>
</tr>
<tr>
<td>E07E</td>
<td>Call Function Handler &lt;E087&gt;</td>
</tr>
<tr>
<td>E081</td>
<td>If no errors then exit &gt;&gt;E086</td>
</tr>
<tr>
<td>E083</td>
<td>Else - call System error handler &lt;BF09&gt;</td>
</tr>
<tr>
<td>E086</td>
<td>Return to caller</td>
</tr>
<tr>
<td>E087</td>
<td>Indirect JUMP to Handler &gt;&gt;FEB5</td>
</tr>
<tr>
<td>E08A</td>
<td>CHECK CALLER'S PATHNAME</td>
</tr>
<tr>
<td>E09A</td>
<td>Set (F48) -&gt; Pathname</td>
</tr>
<tr>
<td>E095</td>
<td>---</td>
</tr>
<tr>
<td>E099</td>
<td>Assume partial Pathname (FEB8)</td>
</tr>
<tr>
<td>E09C</td>
<td>No Pathname in my area yet (D788)</td>
</tr>
<tr>
<td>E09F</td>
<td>Check length of caller's Pathname</td>
</tr>
<tr>
<td>E0A1</td>
<td>Zero is no good &gt;&gt;E0F8</td>
</tr>
<tr>
<td>E0A5</td>
<td>Nor is 65 or more &gt;&gt;E0F3</td>
</tr>
<tr>
<td>E0A7</td>
<td>Save length (FE9E)</td>
</tr>
<tr>
<td>E0AA</td>
<td>Length + 1 (FE9E)</td>
</tr>
<tr>
<td>E0AE</td>
<td>Get first character of his name</td>
</tr>
<tr>
<td>E0B2</td>
<td>Is it &quot;/&quot;?</td>
</tr>
<tr>
<td>E0B4</td>
<td>No &gt;&gt;E0BA</td>
</tr>
<tr>
<td>E0B6</td>
<td>Yes - indicate fully qualified name (FEB8)</td>
</tr>
<tr>
<td>E0BA</td>
<td>Bump past &quot;/&quot;</td>
</tr>
<tr>
<td>E0BB</td>
<td>---</td>
</tr>
<tr>
<td>E0BC</td>
<td>Length of Index level is -1 initially (D788)</td>
</tr>
<tr>
<td>E0BF</td>
<td>First character of Index level (counter)</td>
</tr>
<tr>
<td>E0C2</td>
<td>Start of upcoming Index level in name (FEB8)</td>
</tr>
<tr>
<td>E0C5</td>
<td>At end of name yet? (FE9E)</td>
</tr>
<tr>
<td>E0C8</td>
<td>Yes &gt;&gt;E0FF</td>
</tr>
</tbody>
</table>
ProDOS MLI -- V1.1.1 -- 18 SEP 84

ADDR  DESCRIPTION/CONTENTS

E0CA No - get next character in his name
E0D0 Is it "/"?
E0D2 Yes >> E14
E0D4 No - lower case?
E0D6 No >> E0DA
E0DB Yes - force upper case
E0DA Copy to my Pathname buffer (D703)
E0DD Increment Index level counter (FE88)
E0E0 Subsequent characters may be A-Z,0-9 or . >> E0E7
E0E2 Increment Index level counter (FE88)
E0E5 First character must be alphabetic >> E0F3
E0E7 Is it "/"?
E0E9 Yes - get next character >> E0C5
E0EB No - is it special or control character
E0ED Yes - Bad Pathname then >> E0FB
E0EF Is it numeric?
E0F1 Yes - get next character >> E0C5
E0F3 Is it Alphabetic?
E0F9 If so get next character >> E0C5
E0FB Else
E0FC Bad Pathname
E0FE RETURN
E0FF ---
E101 Any characters in last Index level? (FE88)
E104 Yes >> E10A
E106 No, zero characters in it (FE88)
E109 And toss out last "/
E10A ---
E10B Mark end of name with $00 (D709)
E10E Name too long >> E0FB
E110 No - save final length (FE9E)
E113 Set X to 0
E117 Last Index more than 15 characters?
E119 Yes - then good >> E0FB
E128 Save output index (FE8D)
E129 Last level (FE8A)
E12A Fully qualified name? (FEBC)
E12D Yes >> E134
E12F No - Got a Prefix (BF9A)
E132 No - error >> E0FB
E134 Else, okay to exit

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

E135 Copy Pathname $08A
E138 It's okay >> E144
E13A Check length of Volume name (D700)
E13F If zero - no Prefix wanted (BF9A)
E142 Exit with no error
E143 RETURN
E144 Get File entry for last index $5A3
E147 Okay? >> E14D
E149 Invalid Pathname?
E14B No - Out now! >> E10B
E14D Sub Directory file? (FE5F)
E154 No, error >> E1B9
E156 Fully qualified path? (FEBC)
E159 Yes >> E15E
E15B No - use Old Prefix also (BF9A)
E15E ---
E160 Compute new Prefix Index (FE9E)
E163 Does new Prefix exceed 64 characters?
E165 Yes - Bad Path error >> E0FB
E168 Store new Prefix pointer (BF9A)
E16E Set Device Number of Prefix Directory (FE9F)
E174 Save Keyblock for Prefix Directory (FEA0)
E17D Copy Prefix to top of Path buffer (D700)
E180 (preceded by old Prefix if one exists) (D700)
E188 Exit normally
E189 Bad File Type Error
E18B ---
E18C RETURN

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

E18D Set ($4E) -> Data Buffer
E199 Set Length = 64 (max)
E1A3 Validity check buffer storage ($C2)
E1A6 Error? >> E1B8
E1A9 Get Prefix index (BF9A)
E1AD No Prefix? - Length = 0 >> E1B4
E1B0 Complement for length
E1B4 Store in first byte of buffer
E1B6 If null Prefix exit >> E1CE
E1B8 ---
E1B9 Copy Prefix to caller's buffer replacing (D700)
E1BC index level name length bytes with "/"
<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>E1C6</td>
<td>---</td>
</tr>
<tr>
<td>E1CA</td>
<td>End it with a &quot;/&quot;</td>
</tr>
<tr>
<td>E1CE</td>
<td>---</td>
</tr>
<tr>
<td>E1CF</td>
<td>Exit normally</td>
</tr>
<tr>
<td>E1D0</td>
<td>*************** VALIDITY CHECK REFERENCE NUMBER *************** (PASSED BY CALLER)</td>
</tr>
<tr>
<td>E1D0</td>
<td>Get Reference Number</td>
</tr>
<tr>
<td>E1D4</td>
<td>If zero then no good &gt;&gt;E231</td>
</tr>
<tr>
<td>E1D8</td>
<td>If &gt; 8 then no good &gt;&gt;E231</td>
</tr>
<tr>
<td>E1DA</td>
<td>Save Reference Number</td>
</tr>
<tr>
<td>E1DB</td>
<td>Multiply by 32</td>
</tr>
<tr>
<td>E1E1</td>
<td>Result gives offset into FCB's (FE92)</td>
</tr>
<tr>
<td>E1E5</td>
<td>Get back Reference Number</td>
</tr>
<tr>
<td>E1E6</td>
<td>File Control Block active this Reference? (D806)</td>
</tr>
<tr>
<td>E1E9</td>
<td>No = Bad Reference Number &gt;&gt;E22C</td>
</tr>
<tr>
<td>E1EB</td>
<td>Get Buffer Number (D80B)</td>
</tr>
<tr>
<td>E1EE</td>
<td>Find Buffer address in Global Page &lt;FC3C&gt;</td>
</tr>
<tr>
<td>E1F4</td>
<td>No Buffer? &gt;&gt;E21D</td>
</tr>
<tr>
<td>E1FA</td>
<td>Buffer okay, save Page Pointer in $48</td>
</tr>
<tr>
<td>E1FC</td>
<td>Set last Device used in Global Page (D801)</td>
</tr>
<tr>
<td>E202</td>
<td>Finish setting up pointers (FE0D)</td>
</tr>
<tr>
<td>E205</td>
<td>($4A) -&gt; 1st Block of Buffer (data)</td>
</tr>
<tr>
<td>E207</td>
<td>($48) -&gt; 2nd Block of Buffer (index)</td>
</tr>
<tr>
<td>E209</td>
<td>---</td>
</tr>
<tr>
<td>E20A</td>
<td>Search all Volume Control Blocks (D910)</td>
</tr>
<tr>
<td>E20D</td>
<td>For the one which goes with requested unit (D801)</td>
</tr>
<tr>
<td>E212</td>
<td>---</td>
</tr>
<tr>
<td>E218</td>
<td>Can't find matching Volume Control Block</td>
</tr>
<tr>
<td>E21A</td>
<td>So die with error type $0A &lt;BPMC&gt;</td>
</tr>
<tr>
<td>E21D</td>
<td>No Buffer in open File Control Block</td>
</tr>
<tr>
<td>E21P</td>
<td>So die with error type $0B &lt;BPMC&gt;</td>
</tr>
<tr>
<td>E222</td>
<td>Is Volume mounted? (D900)</td>
</tr>
<tr>
<td>E225</td>
<td>No, keep looking &gt;&gt;E212</td>
</tr>
<tr>
<td>E227</td>
<td>Save Volume Control Block index (FE91)</td>
</tr>
<tr>
<td>E22B</td>
<td>Exit normally</td>
</tr>
<tr>
<td>E22C</td>
<td>---</td>
</tr>
<tr>
<td>E22E</td>
<td>This looks wrong!!! (FE92)</td>
</tr>
<tr>
<td>E231</td>
<td>Bad Reference Number error</td>
</tr>
<tr>
<td>E234</td>
<td>RETURN</td>
</tr>
<tr>
<td>E235</td>
<td>*********************** MLI ONLINE CALL ***********************</td>
</tr>
</tbody>
</table>

---
E2C3 ******* ERROR ******************************
Store code in data buffer entry

E2C4 ---
E2C5 Store Device Number in entry <E2F6>
E2C6 Store error code next
E2C7 Duplicate Volume error?
E2C8 No - done >>E2DF
E2C9 Store Device Number for duplicate next (FE86)
E2CA No Duplicate now
E2CB Exit with error
E2CC RETURN

E2E1 ********** MAKE ONLINE VOLUME ENTRY ***********************
E2E2 Get name length for loop index (D900)
E2E3 Copy name to Buffer entry (D900)
E2E4 Done yet? (FE88)
E2E5 No, do another >>E2EA
E2E6 Yes, find current Buffer entry (FB6A)
E2E7 Store Device number (BF38)
E2E8 Return to caller
E2E9 ---

E301 ********** MLI CREATE CALL ***********************

E302 Follow Path to File <E5B6>
E303 Error? - I'm expecting one >>E309
E304 If File was found - Duplicate error
E305 ---
E306 Return to caller

E307 File not found?
E308 No, then a real error occurred >>E309
E309 Yes, get requested storage type
E310 Is it $0, $01, $02 or $07?
E311 Yes, carry on >>E31B
E312 Is it $0D?
E313 No, then exit with error >>E32B
E314 Get status of this device (BF30)
E315 Exit on error >>E32E
E316 Is there a free Directory entry? (FE9B)
E317 No >>E32F
E318 Yes - continue >>E3C1

E32B Indicate Bad Storage Type
E32C Return to caller
E32D Is this the Volume Directory? (FE46)
E32E No, we can extend it >>E33B
E32F Yes, Indicate Volume Directory Full error
E330 Return to caller

* EXTEND DIRECTORY FILE *

E331 Save old current Block number
E332 Allocate a Block on Disk <EAB6>
E333 Save the number
E334 Replace BLKNUM
E335 Was there a free Block?
E336 No, then exit >>E32E
E337 Yes, set up forward pointer in old one (DCU2)
E338 to point to it (DCU3)
E339 and Write old Directory Block <E5EA>
E33A Error? Yes, then exit >>E32E
E33B Set BLKNUM -> new Block number
E33C Back point to old Directory Block (DC02)
E33D Loop until done >>E35B
E33E Zero remainder of Block Buffer (DC02)
E33F (including forward pointer) (D000)
E340 Loop until done >>E36A
E341 Write new Directory Block <E5EA>
E342 Error? Yes, then exit >>E32E
E343 Set BLKNUM -> Parent Directory number (FE46)
E344 Read Block with my entry <E5EB>
E345 Entry number of my Directory (FE48)
E346 None relocatable!!
E347 Set ($48) -> Buffer
E348 Skip link pointers
E349 ---
E350 Count entries
E351 Skip to next (FE49)
E352 Add 1 to Blocks used
E353 and $200 to EOF mark (FDCE)
E354 A in entry
E355 Loop until done >>E35F
E356 Write back Block to Parent Directory <E5EA>
E357 Error? then exit >>E3C0
E358 Start all over now that there's room >>E302
ProDOS MLI -- V1.1.1 -- 18 SEP 84
NEXT OBJECT ADDR: E3B1

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>E3B4</td>
<td>************ ZERO $F600 ****************************</td>
</tr>
<tr>
<td>E3B4</td>
<td>Zero $F600 Block Buffer</td>
</tr>
<tr>
<td>E3C0</td>
<td>Return to caller</td>
</tr>
<tr>
<td>E3C1</td>
<td>************ BUILD NEW FILE ****************************</td>
</tr>
<tr>
<td>E3C1</td>
<td>Call Zero $F600 routine &lt;E3B4&gt;</td>
</tr>
<tr>
<td>E3C4</td>
<td>Copy Datetime (Creation)</td>
</tr>
<tr>
<td>E3C6</td>
<td>to my variables</td>
</tr>
<tr>
<td>E3D2</td>
<td>Loop until done &gt;&gt;E3C6</td>
</tr>
<tr>
<td>E3D4</td>
<td>Did he give Datetime (Creation)?</td>
</tr>
<tr>
<td>E3D5</td>
<td>Yes, carry on &gt;&gt;E3E2</td>
</tr>
<tr>
<td>E3D7</td>
<td>No, then use</td>
</tr>
<tr>
<td>E3D9</td>
<td>System Datetime instead (BF90)</td>
</tr>
<tr>
<td>E3E2</td>
<td>If Storage type is $00, $01, $02 or $03</td>
</tr>
<tr>
<td>E3E4</td>
<td>force it to $10</td>
</tr>
<tr>
<td>E3E6</td>
<td>else use a $D9</td>
</tr>
<tr>
<td>E3EC</td>
<td>Find File name (FEBA)</td>
</tr>
<tr>
<td>E3EF</td>
<td>OR Storage type to name length (D70W)</td>
</tr>
<tr>
<td>E3F2</td>
<td>Store Type/Length (FE5F)</td>
</tr>
<tr>
<td>E3F5</td>
<td>Isolate name length</td>
</tr>
<tr>
<td>E3F9</td>
<td>Copy File name to File Entry Buffer (FEBA)</td>
</tr>
<tr>
<td>E407</td>
<td>Copy caller's Access Byte</td>
</tr>
<tr>
<td></td>
<td>NOTE: This should be validity checked!!!</td>
</tr>
<tr>
<td>E40F</td>
<td>and copy File type</td>
</tr>
<tr>
<td>E414</td>
<td>---</td>
</tr>
<tr>
<td>E415</td>
<td>and AUX_TYPE</td>
</tr>
<tr>
<td>E41E</td>
<td>Copy Version and Min Version (0,0) (FD80)</td>
</tr>
<tr>
<td>E421</td>
<td>constants to entry (FE70B)</td>
</tr>
<tr>
<td>E42A</td>
<td>Indicate 1 Block used</td>
</tr>
<tr>
<td>E44F</td>
<td>Copy Directory Header Block number (FE5A)</td>
</tr>
<tr>
<td>E43E</td>
<td>Is this a Seedling file?</td>
</tr>
<tr>
<td>E440</td>
<td>Yes &gt;&gt;E479</td>
</tr>
<tr>
<td>E442</td>
<td>No, Directory file - Build Header in $F600</td>
</tr>
<tr>
<td>E444</td>
<td>Copy completed Directory entry (FE5F)</td>
</tr>
<tr>
<td>E447</td>
<td>to $F600 buffer first (DC04)</td>
</tr>
<tr>
<td>E44B</td>
<td>Loop until done &gt;&gt;E444</td>
</tr>
<tr>
<td>E44D</td>
<td>Make Storage type $E in Header itself</td>
</tr>
<tr>
<td>E452</td>
<td>Put &quot;HUSTON&quot; (Author) in Reserved area</td>
</tr>
<tr>
<td>E45A</td>
<td>and Version, Min Version, Access, (DFD0)</td>
</tr>
<tr>
<td>E45D</td>
<td>Entry-length, File count and (DC20)</td>
</tr>
<tr>
<td>E460</td>
<td>Parent pointer from constants</td>
</tr>
<tr>
<td>E461</td>
<td>Loop until done &gt;&gt;E454</td>
</tr>
<tr>
<td>E465</td>
<td>Copy Parent Block entry number (FE5C)</td>
</tr>
<tr>
<td>E46C</td>
<td>Loop until done &gt;&gt;E465</td>
</tr>
<tr>
<td>E46E</td>
<td>Copy Parent entry Length (FE51)</td>
</tr>
<tr>
<td>E476</td>
<td>EDP = $200 (FE75)</td>
</tr>
<tr>
<td>E479</td>
<td>Allocate a new disk block &lt;EAB6&gt;</td>
</tr>
<tr>
<td>E47C</td>
<td>error? &gt;&gt;E485</td>
</tr>
</tbody>
</table>

ProDOS MLI -- V1.1.1 -- 18 SEP 84
NEXT OBJECT ADDR: E47E

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>E47E</td>
<td>Store it in key pointer of entry (FE70)</td>
</tr>
<tr>
<td>E484</td>
<td>and in BLKNUM for I/O</td>
</tr>
<tr>
<td>E48B</td>
<td>Write zeroed (or DIR HDR) key block &lt;EBEA&gt;</td>
</tr>
<tr>
<td>E48B</td>
<td>error? &gt;&gt;E485</td>
</tr>
<tr>
<td>E48D</td>
<td>Bump parent's file count (FE53)</td>
</tr>
<tr>
<td>E495</td>
<td>Go update directory &lt;E4B6&gt;</td>
</tr>
<tr>
<td>E498</td>
<td>error? &gt;&gt;E485</td>
</tr>
<tr>
<td>E49A</td>
<td>Checkpoint Volume Bit Map and exit. &gt;&gt;EB93</td>
</tr>
<tr>
<td>E49D</td>
<td>************ POINT $48/49 AT DIRECTORY ENTRY *******************</td>
</tr>
<tr>
<td>E49D</td>
<td>$48/$49 --&gt; Entry</td>
</tr>
<tr>
<td>E4A1</td>
<td>Skip link pointers (+4)</td>
</tr>
<tr>
<td>E4A3</td>
<td>File entry number counter (FE5E)</td>
</tr>
<tr>
<td>E4A6</td>
<td>---</td>
</tr>
<tr>
<td>E4A7</td>
<td>Skip to proper entry</td>
</tr>
<tr>
<td>E4AA</td>
<td>Add entry length (FE51)</td>
</tr>
<tr>
<td>E4AF</td>
<td>(bump MSB)</td>
</tr>
<tr>
<td>E4B3</td>
<td>(store LSB)</td>
</tr>
<tr>
<td>E4B5</td>
<td>RETURN</td>
</tr>
<tr>
<td>E486</td>
<td>************ UPDATE DIRECTOR(y)s (S) ****************************</td>
</tr>
<tr>
<td>E486</td>
<td>System date available? (BF90)</td>
</tr>
<tr>
<td>E489</td>
<td>no, forget it &gt;&gt;E4C6</td>
</tr>
<tr>
<td>E48D</td>
<td>yes, copy to last modified date field (BF90)</td>
</tr>
<tr>
<td>E4C6</td>
<td>turn on BUBIT (backup) if appropriate (FE7D)</td>
</tr>
<tr>
<td>E4CF</td>
<td>set DEVNUM of parent (FE59)</td>
</tr>
<tr>
<td>E4D5</td>
<td>and BLKNUM (FE5C)</td>
</tr>
<tr>
<td>E4DF</td>
<td>reread DIR block containing entry &lt;EBE6&gt;</td>
</tr>
<tr>
<td>E4E2</td>
<td>error? &gt;&gt;E485</td>
</tr>
<tr>
<td>E4E4</td>
<td>Point to proper entry in buffer &lt;E49D&gt;</td>
</tr>
<tr>
<td>E4EB</td>
<td>Copy constructed entry to buffer (FE5F)</td>
</tr>
<tr>
<td>E4F6</td>
<td>Is this block the DIR HDR block?</td>
</tr>
<tr>
<td>E501</td>
<td>no, write back new entry &lt;EBEA&gt;</td>
</tr>
<tr>
<td>E504</td>
<td>error? &gt;&gt;E485</td>
</tr>
<tr>
<td>E510</td>
<td>and then read DIR HDR block &lt;EBE6&gt;</td>
</tr>
<tr>
<td>E513</td>
<td>error? &gt;&gt;E485</td>
</tr>
<tr>
<td>E515</td>
<td>in any case..</td>
</tr>
<tr>
<td>E517</td>
<td>copy back update file count to HDR (FE53)</td>
</tr>
<tr>
<td>E520</td>
<td>and ACCESS byte (with Backup) (FE58)</td>
</tr>
<tr>
<td>E526</td>
<td>write back HDR block &lt;EBEA&gt;</td>
</tr>
<tr>
<td>E529</td>
<td>error? &gt;&gt;E583</td>
</tr>
<tr>
<td>E52B</td>
<td>is this the VOL DIR? (DC04)</td>
</tr>
<tr>
<td>E532</td>
<td>yes, all done --&gt; exit &gt;&gt;E5A1</td>
</tr>
<tr>
<td>E534</td>
<td>no, subdirectory... (DC27)</td>
</tr>
<tr>
<td>E537</td>
<td>get parent pointer</td>
</tr>
<tr>
<td>E53E</td>
<td>get parent entry no... (DC29)</td>
</tr>
<tr>
<td>E544</td>
<td>and entry len (DC2A)</td>
</tr>
<tr>
<td>E54A</td>
<td>read parent DIR block &lt;EBE6&gt;</td>
</tr>
<tr>
<td>E54D</td>
<td>error? &gt;&gt;E583</td>
</tr>
<tr>
<td>ADDR</td>
<td>DESCRIPTION/CONTENTS</td>
</tr>
<tr>
<td>------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>E54F</td>
<td>find entry for this subdirectory (&lt;E49D&gt;)</td>
</tr>
<tr>
<td>E552</td>
<td>system date available? ((BP90))</td>
</tr>
<tr>
<td>E555</td>
<td>no (\rightarrow) E564</td>
</tr>
<tr>
<td>E557</td>
<td>yes,</td>
</tr>
<tr>
<td>E55B</td>
<td>copy system date/time to... ((BP90))</td>
</tr>
<tr>
<td>E55E</td>
<td>modified date/time in entry</td>
</tr>
<tr>
<td>E564</td>
<td>write it back (&lt;E6B9&gt;)</td>
</tr>
<tr>
<td>E567</td>
<td>error? (\rightarrow) E583</td>
</tr>
<tr>
<td>E56B</td>
<td>BLKNUM = Hdr block number</td>
</tr>
<tr>
<td>E574</td>
<td>same block we have now?</td>
</tr>
<tr>
<td>E578</td>
<td>yes, go back and date stamp (\rightarrow) E52B</td>
</tr>
<tr>
<td>E57A</td>
<td>no,</td>
</tr>
<tr>
<td>E57E</td>
<td>read HDR block (&lt;E6B9&gt;)</td>
</tr>
<tr>
<td>E581</td>
<td>and go back to date stamp parent DIR (\rightarrow) E52B</td>
</tr>
<tr>
<td>E583</td>
<td>error? then exit</td>
</tr>
<tr>
<td>E584</td>
<td>********** NOT ProDos VOLUME ERROR **********</td>
</tr>
<tr>
<td>E587</td>
<td>RETURN</td>
</tr>
<tr>
<td>E588</td>
<td>********** IS THIS ProDos VOLUME? **********</td>
</tr>
<tr>
<td>E58B</td>
<td>Does previous block ptr = (_?) ((DC08))</td>
</tr>
<tr>
<td>E596</td>
<td>no, not a ProDos volume (\rightarrow) E584</td>
</tr>
<tr>
<td>E598</td>
<td>else, ((DC08))</td>
</tr>
<tr>
<td>E59D</td>
<td>does VOL DIR's STORAGE TYPE = $/E or $/F?</td>
</tr>
<tr>
<td>E59F</td>
<td>no, error (\rightarrow) E584</td>
</tr>
<tr>
<td>E5A1</td>
<td>else, ok</td>
</tr>
<tr>
<td>E5A2</td>
<td>RETURN</td>
</tr>
<tr>
<td>E5A3</td>
<td>********** GET FILE ENTRY **********</td>
</tr>
<tr>
<td>E5A3</td>
<td>follow path to it's end (&lt;E5B6&gt;)</td>
</tr>
<tr>
<td>E5A6</td>
<td>error? (\rightarrow) E585</td>
</tr>
<tr>
<td>E5A8</td>
<td>copy file entry</td>
</tr>
<tr>
<td>E5B3</td>
<td>and exit</td>
</tr>
<tr>
<td>E5B5</td>
<td>RETURN</td>
</tr>
<tr>
<td>E5B6</td>
<td>********** FOLLOW PATH TO A FILE **********</td>
</tr>
<tr>
<td>E5B6</td>
<td>get base dir's data (&lt;E73A&gt;)</td>
</tr>
<tr>
<td>E5B9</td>
<td>error? (\rightarrow) E60D</td>
</tr>
<tr>
<td>E5BB</td>
<td>another subdirectory in the path? (\rightarrow) E585</td>
</tr>
<tr>
<td>E5BD</td>
<td>no, at end of path ((E635))</td>
</tr>
<tr>
<td>E5CB</td>
<td>$/B/S49 (\rightarrow) $/F684 ((\text{HDR}))</td>
</tr>
<tr>
<td>E5CB</td>
<td>copy part of HDR to file entry</td>
</tr>
<tr>
<td>E5D2</td>
<td>file type = $/F ((\text{Directory})) ((PDE9))</td>
</tr>
<tr>
<td>E5D5</td>
<td>BLOCK = 2 ((PE5F))</td>
</tr>
<tr>
<td>E5D8</td>
<td>No. blocks used = 4</td>
</tr>
<tr>
<td>E5D9</td>
<td>EOF = $/00</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>E5DD</td>
<td>TYPE = subdirectory (($D0))</td>
</tr>
<tr>
<td>E5E2</td>
<td>return to caller</td>
</tr>
<tr>
<td>E5F4</td>
<td>RETURN</td>
</tr>
</tbody>
</table>

*** SCAN DIRECTORY FOR FILE ***

| E5F5 | indicate no free entry found as yet |
| E5FA | signal in HDR block |
| E5FB | zero count of names examined |
| E5F0 | find name in block \((\text{E6E}9)\) |
| E5F3 | got it! \(\rightarrow\) E65A |
| E5F5 | not yet, how many entries expected? \((PE98)\) |
| E5F8 | less entry no. I just searched \((\text{FE97})\) |
| E5F0 | more file entries left to search? \(\rightarrow\) E60F |
| E608 | no, directory error |
| E60D | --- |
| E60E | RETURN |

| E60F | yes, update entries left counter \((PE98)\) |
| E615 | back to first buffer page \((249)\) |
| E617 | check next block pointer \((DC02)\) |
| E61F | if zero, directory error \(\rightarrow\) E608 |
| E621 | BLKNUM = next directory block |
| E62B | read next block \(<E6B9>\) |
| E62B | no errors, loop back for more \(\rightarrow\) E5EB |
| E62D | exit if error |

*** NO MORE FILE ENTRIES ***

| E62E | free entry found in directory? \((\text{FE9B})\) |
| E631 | yes \(\rightarrow\) E64E |
| E633 | no, check pointers \((\text{DC02})\) |
| E636 | is there another block after this one? \(\rightarrow\) E63D |
| E63B | no... \(\rightarrow\) E64E |
| E63D | yes, free entry will be... \((\text{FES})\) |
| E646 | first in that block |
| E64B | indicate free entry available \((\text{FE9B})\) |
| E64E | find next index name \(<E77B>\) |
| E651 | exiting with error |
| E652 | no more indicies in path, file not found \(\rightarrow\) E657 |
| E654 | else, path not found |
| E656 | RETURN |
| E657 | file not found error |
| E659 | RETURN |

*** FOUND FILE ENTRY ***
E65A advance to next subdir in path <E774>
E65D end -- save entry no. and exit >>E6CB
E661 get type of entry
E665 subdir?
E667 no, bad path then >>E65l
E66B copy key block no...
E66D to BLKNUM
E670 and to current DIR block no (FE5A)
E67A go read key block of subdirectory <EBEE>
E67D error? >>E6A3
E682 new file count (FE98)
E68B check minimum version (DC21)
E69D too new? >>E6A1
E696 count bits in reserved field of DIR hdr
E697 --- >>E69A
E69A ---
E69D there must be 5 bits on (normally $75)
E69F (there are) >>E6A5
E6A1 or else, incompatible file format
E6A3 ---
E6A4 RETURN

E6A5 copy DIR HDR <E6AB>
E6A9 and go scan for next level >>E5E5

E6AB ********** COPY DIRECTORY HDR ********************

E6AB Copy:
E6AD CREATION, VERSION, MIN_VERS, ACCESS, (DC1C)
E6B0 ENTRY LEN, ENTRIES_PER_BLK, FILE_COUNT (FE4A)
E6B6 volume directory? (DC04)
E6BD if so, exit now >>E6CA
E6C1 else, copy PARENT_POINTER, (DC27)
E6C4 PARENT_ENTRY_NO., and PARENT_ENTRY_LEN (FE46)
E6CA RETURN

E6CB ********** SAVE DIR ENTRY NO. & BLOCK ********************

E6CB compute entry number (FE52)
E6D4 save it (FE5E)
E6D9 and the block it's in (FE5C)
E6E2 exit

E6E3 ********** SEARCH ONE DIR BLOCK FOR FILE ***********

E6E3 get entries in this block (FE52)
E6E9 $48/$49 --> first entry (E635)
E6F0 ---
E6F2 skip HDR? >>E727
E6F4 no, non empty entry?
Beneath Apple ProDOS Supplement

ProDOS MLI -- V1.1.1 -- 18 SEP 84

************************** FIND BASE DIRECTORY **************************

E793 ---
E795 get old PFIXPTR (BF9A)
E798 fully qualified pathname? (FEBD)
E79B no >>E79E
E79D yes, no old PFIXPTR anymore
E79E save old prefix index (FEBB)
E7A1 DEVNUM=0 (BF30)
E7A4 ---

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

E7A6 scan (D900)
E7A9 got one >>E7B7
E7BB else, bump to next VCB
E7B4 no mounted vols? remount them >>E808

*** FIND LAST DIR IN PREFIX OR TOL DIR ***

E7B7 store name length (FEB8)
E7BD same name as in pathname? (D700)
E7BD no -- skip it >>E7B8
E7CB save VCB index (FE91)
E7CC DEVNUM = VCB's unit no. (D916)
E7DC get old prefix index (FEBB)
E7DF ---
E7E8 accumulate a new index (FEBD)
E7E3 no previous prefix? >>E7F5
E7E6 find last name in prefix (D700)
E7EB read prefix directory instead of vol dir (FEA0)
E7F5 read block <EBE9>
E7F8 error? >>E800
E7FA is this the right directory? <E89E>
E7FD no >>E800
E7FF yes -- exit!

 *** IF NOT THERE, REMOUNT ALL VOLS ***
 *** AND CHECK THEM ***

E800 open files? (FE91)
E806 yes, give up now >>E821
E808 else, (FEB9)
E80B put back old prefix length (FEBA)
E80E copy DVCLST from global page <E864>
E814 use last device accessed first >>E825
E816 if none, get last in my device table (BF31)
E821 volume not found error
E824 RETURN

ProDOS MLI -- V1.1.1 -- 18 SEP 84

************************** COPY GLBL DEVLST TO MY TABLE **************************

E825 ---
E828 search for device in device table (FECB)
E830 device not found >>E831
E832 when found, make it active device (BF30)
E837 remove it from table (FECB)
E83A find its VCB <EB76>
E83D not found? >>E861
E83F volume mounted there? (FE91)
E845 no >>E84C
E847 yes, open files here? (D911)
E84A yes, skip it -- get next unit >>E816
E84C else,
E84E BLKNUM = 2 (vol dir)
E854 read volume directory <EB34>
E857 error? >>E816
E859 mount volume on VCB <EB34>
E85C error? >>E816
E85E is this his chosen volume? <E89E>
E861 no, try again >>E816
E863 yes, exit

E864 COPILY GLBL DEVLST TO MY TABLE

E864 start with last device (BF31)
E867 get a unit number (BF12)
E86C copy it to device table (FECB)
E872 return count of devices (BF31)
E875 RETURN

E876 SCAN VCB's FOR DEVICE NO.

E876 ---
E87A scan VCB's for a given device number
E881 not it? >>E888
E883 is it, save VCB index (FE91)
E886 and exit normally
E887 RETURN

E888 else, volume mounted here? (D900)
E88B yes >>E891
E88E no, save VCB index to empty unit (FE91)
E891 ---
E893 bump to next VCB
E895 and go look at it >>E87A
E897 not found...
E898 any free entries? if not, error >>E89B
E89A else, all is well -- return empty VCB
E89B VCB table full error
E89D RETURN
E89E ********** COMPARE DIR NAME WITH PATH_LVL ***************

E89E ---
E8A3 check DIR type (DC64)
E8A6 VOL_DIR or SUB(DIR)
E8A8 neither >> E8B1
E8AA yes,
E8AC store len of its name (FE86)
E8AF and go on >> E8B6
E8B1 error exit
E8B2 RETURN

E8B3 compare directory names (DC04)
E8B9 no match? >> E8B1
E8C2 they match! exit
E8C3 RETURN

E8C4 ********** MOUNT NEW VOLUME ***********************

E8C4 volume mounted? (FE91)
E8CA no, continue >> E8B1
E8CC yes, same one as one wanted? (E929)
E8CF if so exit, else fall thru >> E928

E8D1 ********** SET UP VCB FROM VOLDIR ***************

E8D1 zero out VCB
E8DC is this a ProDOS volume? (E588)
E8DF no -- exit >> E928
E8E1 duplicate vol in VCB's? (E94A)
E8E4 yes -- exit with that one instead >> E927
E8E6 get new volume's name length (DC04)
E8E8 add to VCB index (FE91)
E8F0 and copy to VCB name field in empty VCB (DC04)
E8F3 store in VCB name len field (D900)
E8F6 copy DEVNUM to VCB unit field (BF36)
E8F8 copy total blocks to VCB (DC29)
E8F9 copy block no. of vol dir to VCB
E8FB copy bit map block no. to VCB (DC27)
E8FC exit
E8FD RETURN

E929 ********** COMPARE VOL NAMES TO MAKE ***************

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

E929 get length (DC04)
E92E same in VCB? (D900)
E931 no >> E941
E934 yes, add len to VCB index to point at (FE90)
E937 last char of name in VCB (FE90)
ProDOS MLI -- V1.1.1 -- 18 SEP 84

ADDRESS DESCRIPTION/CONTENTS

E989 did we find a free bit? (FE91)
E98B no -- volume full >>E9E1
E9C1 save VCB bitmap block offset (D91C)
E9C4 save free block count in VCB also (FE87)
E9D0 are there enough to satisfy request? (D914)
E9DF yes, exit
E9E0 RETURN

E9E1 volume full error
E9E4 RETURN

E9E5 ********** SCAN AND COUNT BITMAP BLOCKS ***************

E9E5 scan through both buffer pages
E9EC counting one bits <EA12>
E9F7 ---
E9FA found free block already? (FE9B)
E9FD if so -- done >>EA1I
E9FF any blocks found yet? (FE86)
E9A5 no >>EA11
EA07 yes, compute total no. of bitmap blocks <EA22>
EA06 less number remaining (FE9C)
EA0E gives bitmap block with first free bit (FE9B)
EA11 exit

EA12 ********** COUNT ONE BITS IN A BYTE **********************

EA12 shift and...
EA15 count bits that are on (FE86)
EA1D exit when byte goes to zero
EA21 RETURN

EA22 ********** COMPUTE NO. BITMAP BLKS -1 **********************

EA22 get blocks on vol count (-1) (FE91)
EA2E ---
EA2F isolate top nibble of block count
EA30 for bit map block count
EA33 RETURN

EA34 ********** FREE A BLOCK ON DISK ********************

EA34 save MSB (FE9C)
EA37 and LSB
EA3B block number passed too big for (D913)
EA3E volume size? (FE9C)
EA42 yes, error >>EA22
EA45 no, get bit position for block no.
EA4B save it (FE9B)
EA4F divide block no. by 8 (FE9C)
EB04 shift bit pattern, bumping block no. LSB
EB05 until a one bit is found >>EB0A
EB0A then shift it back the way it was
EB0B (with that bit turned off) >>EB0A
EB0D store LSB of block no. (FE86)
EB10 store updated byte back in proper page (FEA4)
EB1D indicate bitmap needs checkpoint
EB25 one less block available in VCB (FE91)
EB3A ---
EB3B return with new block no. (FE86)
EB41 RETURN

EB42 ********* GET NEXT BITMAP BLOCK ***********************

EB42 use blocks of vol to compute (FE91)
EB45 number of blocks in bitmap (D913)
EB4C just scanned last block? (D91C)
EB4F yes, no space >>EB60
EB51 no, get next block (D91C)
EB5A checkpoint old one <EB93>
EB5D go read block >>EB64

EB60 disk full error
EB63 RETURN

EB64 ********* READ BITMAP BLOCK ***********************

EB64 have we read bitmap for this unit yet? (FE91)
EB6D yes >>EB7D
EB6F no, checkpoint bitmap of some other unit <EB93>
EB72 error? >>EB92
EB77 get new bitmap unit no. (D910)
EB7D was bitmap modified? (FEA5)
EB80 yes >>EB87
EB82 no, read it <EB4A>
EB85 error? >>EB92
EB87 save bitmap block offset times 2 (FE91)
EB8A (page number) (D91C)
EB91 exit
EB92 RETURN

EB93 ********* CHECKPOINT VOLUME BITMAP ***********************

EB93 ---
EB94 needs checkpoint? (FEA5)
EB97 no >>EB92
EB99 yes, write it <EB66>
EB9C error? >>EB92
EB9E doesn't need checkpoint now
EBA3 exit
**MLE GET MARK CALL**

EC12 error exit
EC14 RETURN

EC15

*** NEED DIFFERENT DATA BLOCK ***

ECB2B set up to...
ECB3 get old mark (FE92)
ECB4 find its block no. (E92) (D813)
ECB5 compute distance in pages from old mark's (FEAB)
ECB6 block to new mark (FE86)
ECB7 earlier -- need new data block >>ECB6
ECB8 too far forward -- need new block >>ECB6
ECB9 MSB's match? (D814)
ECB10 then mark is still in this block >>ED89

ECB11 check storage (D807)
ECB12 zero? >>ECB7
ECB13 seedling, sapling or tree?
ECB14 no, special handling for DR files >>EDB8
ECB15 stomp on FCB2's mark??? (F300+52)
ECB16 (this should never happen anyway) (D800)
ECB17 and return with bad REPFUN error
ECB18 RETURN

**NEED TO CHANGE DATA BLOCKS***

ECB19 does old index block need dumping? (D808)
ECB20 no >>ECB3
ECB21 yes, do so <<ECB8
ECB22 error? >>ECB2
ECB23 check storage type (FE96)
ECB24 tree file?
ECB25 yes >>ED80
ECB26 no, sapling (FEAC)
ECB27 is position in first index block?
ECB28 no, need master index, subindex and data >>ED46
ECB29 yes, first index, reset flags <<EDAF
ECB30 if so, see if in first block >>ECB3

**SAPLING***

ECB31 no, sapling, read its only index block <<E3B
ECB32 error? >>ECB2
ECB33 set block no. of index block
ECB34 and continue below >>ED2D
ECB35 error exit
ECB36 RETURN
*** TREE FILE/NEED ANOTHER INDEX BLOCK ***

ED00  reset flags <EDAF>
ED03  read master index block <EE3B>
ED06  error? >>ECFE
ED08  make index into block from (FEAC)
ED0B  MSB_of_position/2
ED11  is there a subindex there?
ED13  yes! >>ED20
ED19  no, fall thru to make one

*** GET NEW INDEX BLOCK ***

ED1B  need an index and data block
ED1D  go allocate them >>ED46
ED20  set up block no. of subindex
ED28  read it <EE1D>
ED2B  error? >>ECFE

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

ED2D  make block no. out of position (FEAC)
ED36  use as an index to examine index block
ED38  entry
ED3E  if its zero...
ED42  need new data block
ED46  set flags for what to allocate (FE92)
ED4F  new index block being created?
ED51  zero data block in any case <ED67>
ED54  if not index block that's it >>ED9B
ED56  else,
ED5D  zero out index block I/O buffer
ED64  and continue >>ED89

ED67  ******** ZERO OUT DATA BLK I/O BUFFER ***************

ED67  ---
ED6A  zero both pages of buffer
ED71  ---
ED78  RETURN

ED79  ******** READ FILE DATA BLOCK ***********************

ED79  set block no. LSB
ED7B  copy MSB drom index entry
ED7F  ---
ED81  read new data block <EE04>
ED84  error? >>ED9E
ED86  reset block allocation flags <EDAF>

*** GOT DATA BLOCK WANTED ***

ED89  ---
ED90  save previous mark in my variables (D812)
ED96  set new mark in the PCB (F8AA)
EAA1  (54A/54B --> data block buffer)
EAA3  $4C/$4D --> start of the page in
EAA5  the data block buffer which contains (7EA3)
EAAA  the mark.
EAAE  exit

EDAF  ********* RESET BLOCK ALLOC FLAGS **********************

EDAF  get flags (FE92)
EDB5  turn off low 3 bits (allocate no new
EDB7  blocks to file) (D888)
EDBA  RETURN

EDBB  ********* SET DIR FILE POSITION **********************

EDBB  DIR file?
EDBD  yes! >>EDC4
EDBF  no, bad storage type error
EDC1  go to SYSERR <BF99>
EDC4  else, get page distance (FE86)
EDC7  make it into blocks (divide by 2)
EDCE  new position beyond old? (FEAB)
EDD1  yes >>EDE1
EDD3  else, use previous mark
EDDE  copy to BLKNUM <EDEF>
EDD8  error? >>EDFE
EDDA  count its (FE9A)
EDDD  more to skip? >>ED33
EDDF  no, got it >>ED9B
EDE1  use next_block pointer in DIR block
EDE3  copy to BLKNUM <EDEF>
EDE6  error? >>EDFE
EDE8  count its (F9A)
EDEB  more to skip >>EDE1
EDED  got it now! >>ED9B

*** COPY LINK TO BLKNUM ***

EDEF  copy block number link
EDF1  to BLKNUM
EDF4  if non zero,
EDFA  then go read block. >>ED00
EDFC  else, EOF error
EDFE  ---
EDFF  RETURN
ProDOS MLI -- V1.1.1 -- 18 SEP 84
NEXT OBJECT ADDR: EDFF
ADDRES DESCRIPTION/CONTENTS

---
EE00 ****** READ FILE BLOCK ****************************
EE00 set block number to read
EE04 store read 1/0 command
EE08 read to $48/$49 buffer
EE0A read the block <EE61>
EE0D error? >>EE1C
EE12 copy block no. just read to PCB
EE1C exit

EE1D ****** READ SUB-INDEX BLOCK ********************
EE1D set read 1/0 command
EE21 read to $48/$49 buffer
EE23 read the block <EE61>
EE26 error? >>EE36
EE2B save BLKNUM in PCB as current index
EE2D block. (D80E)
EE36 exit

EE37 ****** WRITE KEY INDEX BLOCK ********************
EE37 set write 1/0 command
EE39 and go do the 1/0 >>EE3D

EE3D ****** READ KEY INDEX BLOCK ********************
EE3B set read 1/0 command
EE3D common code, save command
EE40 block no. is key block in PCB (FE92)
EE45 use $48/$49 buffer

*** 1/0 BLOCK ***
EE47 set 1/0 command
EE49 and block no. (D80E)
EE53 must be non-zero block number
EE57 or horrible death!
EE5C fall through to read/write block (D801)

*** SET UP AND DO FILE BLOCK 1/0 ***
EE61 (xreg = buff ptr in zero page)
EE62 disable
EE63 set up buffer pointer
EE6E get DEVNUM from PCB (D801)
EE74 set 1/0 transfer has occurred flag
EE79 set unit no. from DEVNUM (BF30)
EE7E no errors have occurred yet
EE83 do block 1/0 <DEDA>

---
EE86 error? >>EE8B
EE8B no, exit normally
EE8A RETURN

EE8D else, exit with error
EE8D RETURN

EE8E ****** CHECKPOINT BITMAP & KEY BLOCK *******
EE8E checkpoint bitmap buffer <EE93>
EE91 go write key block for file >>EE37

EE94 ****** CHECKPOINT DATA BLOCK BUFFER *******
EE94 buffer pointer at $4A/$4B
EE96 point to block no. in PCB
EE9E go write buffer to disk <EE47>
EEA1 error? >>EEC5
EEA5 go turn off $40 flag in PCB and exit >>EEBC

EEA8 ****** CHECKPOINT INDEX BLOCK BUFFER *******
EEA8 checkpoint volume bitmap <EE93>
EEA9 use $48/$49 buffer
EEAD block no. is current index block in PCB
EEB3 set to write
EEB5 go write it to disk <EE47>
EEB8 error? >>EEC5
EEBC no longer needs checkpoint.
EEBD set flags accordingly (FE92)
EEC5 and exit

EEC6 *********************** MLI OPEN CALL ******
*****************************
EEC6 search path for file <E5A3>
EEC9 found it? >>EECF
EEC8 no, bad path error
EECD exit >>EED6
EECF else, see if PCB already open on file <EF33>
EED2 for write. if not, continue. >>EED6
EED4 else, file already open error
EED6 ---
EED7 RETURN

EED8 get PCB index (FE92)
EEDF free PCB found? >>EED4
EEBF no, all PCB's in use error
EEBE RETURN
ProDOS MLI -- V1.1.1 -- 18 SEP 84

ADDRESS DESCRIPTION/CONTENTS

---

EEE4 zero out unused PCB
EEEF copy file ID fields to PCB
EEF2 (DEVNUM, DIR HDR BLK, DIR BLK, (FE92)
EEF5 DIR ENTRY NO.
EF00 isolate storage type (FE5F)
EF05 and copy to PCB (DB87)
EF08 get access (FE7D)
EF10 DIR file?
EF12 no >>EF16
EF14 yes, we are only reading (I hope)
EF16 update access flag in PCB (DB89)
EF1B write protected? >>EF22
EF1D no, another PCB open on this file? (FE97)
EF20 yes, no touchie >>EF64
EF22 This line left over from version 1.0.11 (FE7C)
EF27 Now always jumps over error exit. >>EF2D
EF29 if bad, unsupported version error
EF2C RETURN

EF2D storage type must be < 54
EF31 or equal to 8D
EF33 else, compatibility error >>EF29
EF35 ---
EF37 copy key block, blocks used, and
EF39 EOF mark to PCB (FE92)
EF49 BLKNUM key block number
EF4E store REFNUM in PCB (FE9A)
EF54 go check and assign I/O buffer <FBED>
EF57 error? >>EF7D
EF59 go find VCB and set buff ptrs <E1EB>
EF5C set current level in PCB (BF94)
EF62 seedling, sapling or tree? (DB87)
EF67 no, skip next stuff >>EF94
EF69 yes, make current mark in PCB outside
EF6B first index block to force a read of all (DB14)
EF6E index blocks and BLOCK 0.
EF72 zero mark wanted, however (FEAA)
EF78 go set mark to zero <EC48>
EF7B ok? >>EF99
EF7D no, save the error code
EF81 got and I/O buffer? (DB8B)
EF84 no >>EF8C
EF86 yes, free it <PC4A>
EF8C mark PCB not in use
EF92 exit with error
EF93 RETURN

---

EF94 else, read key block to I/O buffer <E0E4>
EF97 error? >>EF7D
EF99 bump open file count in VCB (FE91)
EF9F indicate files are open in VCB (DB91)
EFA7 put REF NUM in caller's parmlist (FE92)
EFB1 exit with no errors
EFB2 RETURN

EFB3 ********* FIND A PCB ********************************************

EFB3 clear flags and index byte
EFBE ---
EFBF found a free PCB yet? (FE9J)
EFCC yes >>EFC7?
EFC4 no, bump entry count (FE9A)
EFC7 PCB in use? (DB90)
EFC9 yes >>EPD9
EFC0 no,
EFCF save index to free PCB (FE92)
EFD2 flag that we found one
EFD7 and skip this PCB >>EFF7
EFD9 ---
EFD5 compare file ID's to see if this PCB (DB90)
EFD2 is open on the requested file. (FE58)
EFD5 no match? >>EFF7
EFD9 indicate PCB already open on file (FE97)
EFD8 write enabled? (DB99)
EFD3 if not, allow multiple open access to file >>EFF7
EFD5 else, error exit
EFD6 RETURN

EFD7 return index to start of PCB
EFD9 bump to next PCB
EFD7 and loop >>EFE8
EFD8 when done, exit normally
F000 RETURN

---

F001 ********************************************

***************** MLI READ CALL ********************

F001 point to data buffer <F200>
F004 copy request length <F3F2>
F007 save access
F008 set up marks <F21F>
F00C read access permitted?
F00E yes >>F014
F010 no, access error
F014 will we read past EOF? >>F038
F016 yes, (FE92)
### ProDOS MLI — V1.1.1 — 18 SEP 84

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>F019</td>
<td>LENGTH = EOF - current mark (DB15)</td>
</tr>
<tr>
<td>F031</td>
<td>are we already at EOF? (FEDA)</td>
</tr>
<tr>
<td>F034</td>
<td>no &gt;&gt;F046</td>
</tr>
<tr>
<td>F036</td>
<td>yes, EOF error</td>
</tr>
<tr>
<td>F03B</td>
<td>else, zero length request? (FEDA)</td>
</tr>
<tr>
<td>F041</td>
<td>no &gt;&gt;F046</td>
</tr>
<tr>
<td>F043</td>
<td>yes, set mark and exit &gt;&gt;F059</td>
</tr>
<tr>
<td>F046</td>
<td>validity check data buffer &lt;FC62&gt;</td>
</tr>
<tr>
<td>F049</td>
<td>no good? &gt;&gt;F038</td>
</tr>
<tr>
<td>F04B</td>
<td>ok, get storage type for file &lt;F218&gt;</td>
</tr>
<tr>
<td>F04E</td>
<td>standard kind of file?</td>
</tr>
<tr>
<td>F050</td>
<td>yes &gt;&gt;F055</td>
</tr>
<tr>
<td>F052</td>
<td>no, DIR file &gt;&gt;F1BB</td>
</tr>
<tr>
<td>F055</td>
<td>else, set mark (to read proper buffers) &lt;EC48&gt;</td>
</tr>
<tr>
<td>F05B</td>
<td>error? &gt;&gt;F038</td>
</tr>
<tr>
<td>F05A</td>
<td>set up buffer indexing &lt;F110&gt;</td>
</tr>
<tr>
<td>F05D</td>
<td>move all that can be moved out of data buff &lt;F13A&gt;</td>
</tr>
<tr>
<td>F06B</td>
<td>newline or len=0: exit now &gt;&gt;F043</td>
</tr>
<tr>
<td>F062</td>
<td>newline enabled? continue block by block &gt;&gt;F055</td>
</tr>
<tr>
<td>F064</td>
<td>at least 1 block's worth left to be read? (FEE4)</td>
</tr>
<tr>
<td>F066</td>
<td>if not, never mind &gt;&gt;F055</td>
</tr>
<tr>
<td>F068</td>
<td>if so, store block count wanted (FEAF)</td>
</tr>
<tr>
<td>F06D</td>
<td>get FCB flags &lt;F069&gt;</td>
</tr>
<tr>
<td>F070</td>
<td>data block modified?</td>
</tr>
<tr>
<td>F072</td>
<td>yes, continue block by block for now &gt;&gt;F055</td>
</tr>
</tbody>
</table>

**FAST DIRECT READ ROUTINE**

| F074 | signal no read occurred yet (FEB2) |
| F077 | read directly into caller's data buffer |
| F07P | set mark/read data block to caller's buff <EC48> |
| F082 | error? >>F0ED |
| F084 | bump buffer pointer to next location |
| F08B | drop length remaining by 512 bytes (FEAE) |
| F08E | bump mark (FEAB) |
| F096 | and mark's MSB as necessary (FEAC) |
| F099 | check if we are out of index block (FEAC) |
| F09P | drop counter of multi-blocks (FEAF) |
| F0A2 | and keep on >>F0B1 |
| F0A4 | end of multi-block read, put ptrs back <F1AD> |
| F0A7 | more to read? (FEAD) |
| F0AD | no, exit through finish-up >>F0F9 |
| F0AP | yes, conventional block by block read then >>F055 |

**I/O FINISH UP**

| F0F9 | ********** I/O FINISH UP *********************************************** |
| F0F9 | --- |
| F0FC | return actual length read in caller's list (FEDA) |
| F10D | and exit by setting new mark >>EC48 |

**SET UP BUFFER INDEXING**

| F110 | --- |
| F114 | back up pointer to data buffer by an |
| F116 | amount equal to the LSB of the mark (FEAA) |
| F119 | (which makes indexing easier) |
| F11F | newline mode enabled? (DB1F) |
| F123 | no, CLC >>F12F |
| F125 | yes, SEC |
| F126 | copy newline mask (FEB1) |
| F129 | and newline character (DB8A) |
| F12F | first char index is LSB of mark in YREG (FEAA) |
| F132 | $4C/$4D --- page containing mark |
| F136 | request count LSB in XREG (FEAD) |
| F139 | exit |
**F13A** ********** COPY FROM I/O BLOCK BUFF ***********************

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

EXIT 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

F13A ---

F13B partial page to move? >>F145

F13D no, any full pages left? (FEAB)

F140 no, read complete >>F194

F142 yes, drop MSB of request length (FEAB)

F145 ---

F146 copy one byte 4C --> 4E

F148 end of requested chunk? >>F168

F14D no, newline enabled? >>F17D

F14F ---

F151 no, loop for more >>F146

F153 end of page, bump pointers

F157 bump new mark (FEAB)

F15F finished first page of block buffer?

F163 if so, continue >>F146

F166 no, need another block from disk >>F197

F168 another page in request length? (FEAB)

F16B no >>F187

F16E more in this block-page? >>F176

F170 no, on last page of block?

F174 no >>F179

F176 yes, drop request len by one page (FEAB)

F179 back up to next byte again

F17A go copy next page >>F14D

F17D check for newline

F185 not it, never mind! >>F14F

F187 else, were we done with page?

F188 no >>F194

F18A yes, bump pointer

F18C and mark (FEAB)

F194 set overflow flag (read completed) (F1AC)

F197 update mark LSB (FEAB)

F19C bump request count if necessary

F19D update count LSB (FEAD)

F1A3 point beyond data in caller's buffer

F1AB ---

F1AC and exit

---

**F1AD** ********** CLEANUP AFTER DIRECT I/O **********************

F1AD restore caller's data buffer pointer

F1BE go set buffers/find VCB and exit >>E1EB

**F1BB** ********** DIRECTORY FILE READ **********************

F1BB set mark/read <EC48>

F1BE error? >>F1EF

F1CB set up buffer indexing <F10>

F1C3 move data from I/O buffer <<F13A>

F1C6 need next block? >>F1BB

F1CB no, finish up I/O <<F0F9>

F1CB ok? exit >>F1ED

F1CB not ok, EOF error?

F1DB no, out now >>F1EE

F1D2 yes, point beyond EOF anyway? <ED89>

F1D5 zero out data block I/O buffer <<ED7>

F1DD dummy up an empty DIR block with previous (D810)

F1EE pointer and no forward pointer in I/O

F1EE buffer.

F1E4 zero out current block no. (D810)

F1ED return to caller

F1EE RETURN

F1EF finish up and error exit >>F0F2

**F1F2** ********** COPY CALLER'S I/O LENGTH ********************

F1F2 copy request length to LENGTH and

F1F4 a temporary variable

F205 pick up ACCESS flags for file (FE92)

F20B exit to caller

F20C RETURN

**F20D** ********** POINT $4E/$4F TO CALLER'S ********************

****** DATA BUFFER ******

F20D set up pointer

F218 YREG --> FCB (FE92)

F218 AREG = storage type (D807)

F21E exit

**F21F** ********** COPY FILE MARK AND COMPUTE ********************

****** AND COMPARE END MARK ******
ProDOS MLI -- V1.1.1 -- 18 SEP 84  NEXT OBJECT ADDR: F21F

ADDR   DESCRIPTION/CONTENTS
--------  -----------------------------------------------------
F21F   ---
F22F   copy file mark (D812)
F22B   and set previous mark also (FE8D)
F22E   add length giving new mark in scratch area (FE8A)
F235   (3 byte addition)
F23D   will new mark exceed EOF? (FE86)
F24B   return with carry set accordingly

F24C ******** SET NEW MARK & EOF  ***********************
F24C   set up indexes <F27E>
F25F   set new EOF in FCB (FE8A)
F255   and new mark (FE8D)
F25B   save new mark in scratch variable too (FE86)
F262   does mark exceed EOF? <F27E>
F265   if so, we must extend EOF <F23D>
F268   save old EOF (D815)
F273   set new EOF to mark if necessary (FE86)
F279   ---
F27D   exit

F275 subroutine to set 3 byte indexes
F285   RETURN

F286 ********************** MLi WRITE CALL **********************

F286   copy request length <F1P2>
F28A   copy file mark <F21F>
F28D   extend EOF if needed <F268>
F291   write access enabled?
F293   yes >>F299
F295   no, access error
F299   check status of this device <F45B>
F29C   error? >>F29D
F29E   request length = 0? (FE8A)
F2A4   no >>F2A9
F2A6   yes, exit through finish-up >>F9F9

F2A9   find caller's data buffer <F20D>
F2AC   check storage type
F2AE   if DIR file, error >>F295
F2B0   set mark/read blocks <E048>
F2B3   error? >>F29D
F2B5   get FCB flags <F606>
F2BB   any new blocks needed?
F2BA   no >>F31E
F2BC   yes, allocating them
F2BE   ---

ProDOS MLI -- V1.1.1 -- 18 SEP 84  NEXT OBJECT ADDR: F2BF

ADDR   DESCRIPTION/CONTENTS
--------  -----------------------------------------------------
F2BF   count number of blocks needed
F2C2   store number needed (FE94)
F2CB   see if the blocks are available <E973>
F2CB   no, disk full >>F2D9
F2CD   yes, FCB flags <F606>
F2D0   master index block needed?
F2D2   no >>F2E1
F2D4   yes, go add it <F399>
F2D7   and go on if no errors >>F2ED
F2D9   error,
F2DA   set new mark/EOF <F24C>
F2DE   and finish I/O, exit with error >>F6F2

F2EL   check FCB flags again <F606>
F2E4   need sub-index block?
F2E6   no >>F2ED
F2EB   yes, go do it <F3E4>
F2EB   error? >>F2D9
F2ED   buy a new block for data <F43B>
F2F0   error? >>F2D9
F2F2   get FCB flags <F606>
F2F5   indicate index buffer changed
F2F7   no new blocks needed now
F2F9   update FCB flags (D808)
F2FF   make index block offset from mark
F307   store new block no. in index block (FE87)
F314   and store it as current data block (FE92)
F31E   set up buffer indexing <F118>
F321   start writing <F329>
F324   go see if more blocks are needed >>F2B0
F326   I/O finish up when done >>F8F9

F329 ********** COPY WRITE DATA TO I/O BLOCK ******************

F329   ---
F32C   lower request count by 1 (FE8A)
F334   ---
F335   copy partial page from caller's data
F337   to I/O block buffer
F33C   ---
F33F   next page in caller's area
F343   bump mark by $100 (FE8A)
F34B   still in same I/O block page?
F34F   yes >>F334
F352   no, clear overflow (I/O incomplete) >>F379
<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>F354</td>
<td>any complete pages left to write? (PBAE)</td>
</tr>
<tr>
<td>F357</td>
<td>no &gt;&gt;F369</td>
</tr>
<tr>
<td>F359</td>
<td>yes, more in this page?</td>
</tr>
<tr>
<td>F35A</td>
<td>yes &gt;&gt;F362</td>
</tr>
<tr>
<td>F35C</td>
<td>no, first block-paged?</td>
</tr>
<tr>
<td>F35E</td>
<td>no &gt;&gt;F353</td>
</tr>
<tr>
<td>F362</td>
<td>yes, one less complete page to do (PBAE)</td>
</tr>
<tr>
<td>F365</td>
<td>readjust index</td>
</tr>
<tr>
<td>F366</td>
<td>continue with full page &gt;&gt;F35C</td>
</tr>
<tr>
<td>F369</td>
<td>---</td>
</tr>
<tr>
<td>F36A</td>
<td>a few bytes left to write? &gt;&gt;F376</td>
</tr>
<tr>
<td>F36C</td>
<td>no, bump data buffer by $100</td>
</tr>
<tr>
<td>F36E</td>
<td>and mark (PBAE)</td>
</tr>
<tr>
<td>F371</td>
<td>set overflow (I/O complete) (FIAC)</td>
</tr>
<tr>
<td>F372</td>
<td>store LS6 of mark (PBAE)</td>
</tr>
<tr>
<td>F373</td>
<td>and of request count (FEAD)</td>
</tr>
<tr>
<td>F373</td>
<td>indicate data block modified &lt;F606&gt;</td>
</tr>
<tr>
<td>F363</td>
<td>and DIR entry needs update</td>
</tr>
<tr>
<td>F369</td>
<td>advance pointer into caller's buffer (PBAE)</td>
</tr>
<tr>
<td>F394</td>
<td>set PCB flag to indicate write occurred &lt;F666&gt;</td>
</tr>
<tr>
<td>F398</td>
<td>exit</td>
</tr>
</tbody>
</table>

**F399 ADD NEW MASTER INDEX BLOCK **********************

(MAKE A TREE FILE)

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>F399</td>
<td>add higher level &lt;F3F1&gt;</td>
</tr>
<tr>
<td>F39C</td>
<td>error? &gt;&gt;F380</td>
</tr>
<tr>
<td>F39E</td>
<td>get storage mark</td>
</tr>
<tr>
<td>F3A1</td>
<td>tree?</td>
</tr>
<tr>
<td>F3A3</td>
<td>yes &gt;&gt;F3AA</td>
</tr>
<tr>
<td>F3A5</td>
<td>no, add another level &lt;F3F1&gt;</td>
</tr>
<tr>
<td>F3A8</td>
<td>error? &gt;&gt;F3F0</td>
</tr>
<tr>
<td>F3AA</td>
<td>buy another block &lt;F38B&gt;</td>
</tr>
<tr>
<td>F3AD</td>
<td>error? &gt;&gt;F3F0</td>
</tr>
<tr>
<td>F3AF</td>
<td>male offset into current index block (FEAC)</td>
</tr>
<tr>
<td>F3B2</td>
<td>from current mark</td>
</tr>
<tr>
<td>F3B4</td>
<td>point index to new block (F606)</td>
</tr>
<tr>
<td>F3C3</td>
<td>also save as current data block (FE92)</td>
</tr>
<tr>
<td>F3CD</td>
<td>checkpoint bitmap &amp; key block &lt;E666&gt;</td>
</tr>
<tr>
<td>F3DE</td>
<td>error? &gt;&gt;F3F0</td>
</tr>
<tr>
<td>F3D5</td>
<td>zero out new index block</td>
</tr>
<tr>
<td>F3D6</td>
<td>---</td>
</tr>
<tr>
<td>F3E3</td>
<td>and exit</td>
</tr>
</tbody>
</table>

**F3E4 ADD NEW INDEX BLOCK **********************

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>F3E4</td>
<td>check storage type &lt;F218&gt;</td>
</tr>
<tr>
<td>F3E5</td>
<td>seedling? &gt;&gt;F3F1</td>
</tr>
<tr>
<td>F3E5</td>
<td>no, read index block &lt;E666&gt;</td>
</tr>
<tr>
<td>F3EE</td>
<td>and go add data block &gt;&gt;F3AA</td>
</tr>
<tr>
<td>F3F0</td>
<td>exit if error occurs</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>F3F1</td>
<td>buy a block &lt;F438&gt;</td>
</tr>
<tr>
<td>F3F4</td>
<td>error? &gt;&gt;F437</td>
</tr>
<tr>
<td>F3F9</td>
<td>save old key block number (D66C)</td>
</tr>
<tr>
<td>F401</td>
<td>make new block the key block (D66C)</td>
</tr>
<tr>
<td>F40E</td>
<td>and current index block in PCB (D66F)</td>
</tr>
<tr>
<td>F417</td>
<td>store pointer to old key block</td>
</tr>
<tr>
<td>F41A</td>
<td>in first position of new index</td>
</tr>
<tr>
<td>F421</td>
<td>checkpoint bitmap and new key block &lt;E666&gt;</td>
</tr>
<tr>
<td>F424</td>
<td>error? &gt;&gt;F437</td>
</tr>
<tr>
<td>F426</td>
<td>get storage type &lt;F218&gt;</td>
</tr>
<tr>
<td>F42B</td>
<td>upgrade it to next higher type (D667)</td>
</tr>
<tr>
<td>F42E</td>
<td>indicate DIR entry needs update (D668)</td>
</tr>
<tr>
<td>F42F</td>
<td>exit</td>
</tr>
</tbody>
</table>

**F438 BUY A DISK BLOCK **********************

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>F438</td>
<td>allocate a disk block &lt;E666&gt;</td>
</tr>
<tr>
<td>F43B</td>
<td>error? &gt;&gt;F457</td>
</tr>
<tr>
<td>F43D</td>
<td>get PCB flags &lt;F606&gt;</td>
</tr>
<tr>
<td>F440</td>
<td>indicate DIR entry needs update</td>
</tr>
<tr>
<td>F449</td>
<td>add 1 to blocks in use for file</td>
</tr>
<tr>
<td>F456</td>
<td>---</td>
</tr>
<tr>
<td>F457</td>
<td>exit</td>
</tr>
</tbody>
</table>

**F458 DO STATUS IF NO I/O YET **********************

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>F458</td>
<td>get PCB flags &lt;F606&gt;</td>
</tr>
<tr>
<td>F45B</td>
<td>any buffers in use? (I/O activity)</td>
</tr>
<tr>
<td>F45D</td>
<td>if so, assume its ok &gt;&gt;F456</td>
</tr>
<tr>
<td>F45F</td>
<td>no, (D661)</td>
</tr>
<tr>
<td>F462</td>
<td>select new device (BF30)</td>
</tr>
</tbody>
</table>

*** STATUS CALL ***

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>F465</td>
<td>Save Unit Number</td>
</tr>
<tr>
<td>F467</td>
<td>Save Block Number on stack</td>
</tr>
<tr>
<td>F46D</td>
<td>Indicate Status call</td>
</tr>
<tr>
<td>F471</td>
<td>Indicate Block 0</td>
</tr>
<tr>
<td>F475</td>
<td>Go do I/O &lt;DEDAB&gt;</td>
</tr>
<tr>
<td>F478</td>
<td>Restore Block Number to original value</td>
</tr>
<tr>
<td>F480</td>
<td>Exit</td>
</tr>
</tbody>
</table>
ProDOS MLI -- V1.1.1 -- 18 SEP 84
NEXT OBJECT ADDR: F481

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

F481 *****************************************************************************
****** MLI CLOSE CALL ******
*****************************************************************************

F481 check REF NUM
F485 specific close? >>F48C

*** CLOSE ALL OPEN FILES ***

F487 no errors yet (FEBE)
F48C store FCB index (FE92)
F490 get its level (D81B)
F493 if below system LEVEL, skip it (BF94)
F496 yes, skip it >>F4AD
F498 no, active FCB? (D800)
F49B no >>F4AD
F49D yes, flush it and update directory <<F51E>
F4AD error? >>F4EF
F4A2 no, close specific FCB <<F4C1>
F4A7 is this a close-all?
F4A9 yes, ignore errors >>F4AD
F4AB no, stop on error >>F4EF
F4AD bump FCB index to next one (FE92)
F4B3 and continue >>F48C
F4B5 when done, load error number (FEBE)
F4BB and exit

*** CLOSE SPECIFIC FILE ***

F4BC flush it <<F526>
F4BF error? >>F4EF
F4C1 get buffer number (FE92)
F4C7 free its pages <<FC4A>
F4CA error? >>F48F
F4CC release FCB
F4D4 set DEVMNUM (D801)
F4DA find VCB for device <<E876>
F4DD decrement count of open files in VCB (FE91)
F4E3 some are open... >>F4ED
F4E5 if all are closed, turn off (D911)
F4EB "files open" flag
F4ED ---
F4EE exit
F4EF jump to handle close error >>F5F7

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

F4F2 *****************************************************************************
****** MLI FLUSH CALL ******
*****************************************************************************

F4F2 flush specific file?
F4F6 yes >>F526
F4F8 no, clear flush-all error code (FEBE)
F4FB do all PCBs
F4FD set FCB index for next FCB (FE92)
F501 is this file open? (D80E)
F504 no >>F50B
F506 yes, flush it <<F51E>
F509 error? >>F51B
F50B bump to next FCB (FE92)
F511 and go flush it too >>F4FD
F513 ---
F514 return with error code if any (FEBE)
F51A RETURN
F51B ---

F51E ****** FLUSH A FILE & UPDATE DIRECTORY ******

F51E find buffer/VCB <<E1E8>
F521 no error? >>F530
F523 error = exit >>F5F7

F526 zero out close-all error
F52B validity check REF NUM <<E1D0>
F52E error? >>F51B
F530 is write access allowed? (D809)
F535 no, exit >>F513
F537 has a write occured since last flush? (D81C)
F53A yes >>F543
F53C no, <<F686>
F53F does anything need flushing anyway?
F541 no, then exit now >>F513
F543 else, get FCB flags <<F686>
F546 has data buffer changed?
F548 no >>F54F
F54A yes, checkpoint it <<E994>
F54D error? >>F51B
F54F get flags again <<F606>
F552 has index buffer changed?
F554 no >>F55B
F556 yes, checkpoint it <<E9A8>
F559 error? >>F51B
F55B ---
F562 copy file identifier data to my variables (D800)
F56C set DEVMNUM (BF30)
<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>F56F</td>
<td>BLKNUM = current DIR block (FE5A)</td>
</tr>
<tr>
<td>F579</td>
<td>read DIR block &lt;EBEE&gt;</td>
</tr>
<tr>
<td>F57C</td>
<td>error? &gt;&gt;F518</td>
</tr>
<tr>
<td>F57E</td>
<td>copy directory header &lt;E6AB&gt;</td>
</tr>
<tr>
<td>F591</td>
<td>are we in block with this file's entry? (FE5C)</td>
</tr>
<tr>
<td>F58A</td>
<td>no &gt;&gt;F591</td>
</tr>
<tr>
<td>F58F</td>
<td>yes &gt;&gt;F590</td>
</tr>
<tr>
<td>F591</td>
<td>no, set new block number</td>
</tr>
<tr>
<td>F595</td>
<td>read it &lt;EBEE&gt;</td>
</tr>
<tr>
<td>F59B</td>
<td>point at directory entry in block &lt;E49D&gt;</td>
</tr>
<tr>
<td>F59B</td>
<td>copy file entry from directory &lt;E5AB&gt;</td>
</tr>
<tr>
<td>F5A1</td>
<td>copy blocks used count to entry (D818)</td>
</tr>
<tr>
<td>F5AF</td>
<td>copy new EOF (D815)</td>
</tr>
<tr>
<td>F5B0</td>
<td>and new key block no. (D66C)</td>
</tr>
<tr>
<td>F5C3</td>
<td>isolate new storage type (D865)</td>
</tr>
<tr>
<td>F5CD</td>
<td>combine it with same length (FE5F)</td>
</tr>
<tr>
<td>F5DF</td>
<td>and update type/len field in entry (FE5F)</td>
</tr>
<tr>
<td>F5DB</td>
<td>write entry back to directory &lt;E48B&gt;</td>
</tr>
<tr>
<td>F5D8</td>
<td>error? &gt;&gt;F57</td>
</tr>
<tr>
<td>F5D8</td>
<td>turn off &quot;write occurred&quot; flag (D81C)</td>
</tr>
<tr>
<td>F5E8</td>
<td>same bitmap in memory (FE59)</td>
</tr>
<tr>
<td>F5E5</td>
<td>no, exit now &gt;&gt;F5F5</td>
</tr>
<tr>
<td>F5F0</td>
<td>yes, checkpoint it also &lt;EB93&gt;</td>
</tr>
<tr>
<td>F5F5</td>
<td>no errors, exit</td>
</tr>
<tr>
<td>F5F6</td>
<td>RETURN</td>
</tr>
</tbody>
</table>

F5F7 **************************** CLOSE ERROR ****************************

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>F5F7</td>
<td>is this a close or flush all?</td>
</tr>
<tr>
<td>F5FC</td>
<td>no &gt;&gt;F604</td>
</tr>
<tr>
<td>F600</td>
<td>yes, save error code (FEBE)</td>
</tr>
<tr>
<td>F603</td>
<td>RETURN</td>
</tr>
<tr>
<td>F604</td>
<td>else, real error right now</td>
</tr>
<tr>
<td>F605</td>
<td>RETURN</td>
</tr>
</tbody>
</table>

F606 **************************** GET FCB FLAGS ****************************

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>F606</td>
<td>load FCB flags (FE92)</td>
</tr>
<tr>
<td>F609</td>
<td>from FCB (D8BB)</td>
</tr>
<tr>
<td>F60C</td>
<td>and exit</td>
</tr>
</tbody>
</table>

F60D **************************** FILE ACCESS ERROR ****************************

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>F60D</td>
<td>exit with file access error code</td>
</tr>
<tr>
<td>F610</td>
<td>RETURN</td>
</tr>
</tbody>
</table>

F611 **************************** MLI SET EOF CALL ****************************

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>F611</td>
<td>get storage type &lt;F218&gt;</td>
</tr>
<tr>
<td>F614</td>
<td>if DIR file...</td>
</tr>
<tr>
<td>F616</td>
<td>its an access error &gt;&gt;F60D</td>
</tr>
<tr>
<td>F618</td>
<td>else, save type for truncate to</td>
</tr>
<tr>
<td>F619</td>
<td>mess with.</td>
</tr>
<tr>
<td>F61F</td>
<td>write access permitted? (D809)</td>
</tr>
<tr>
<td>F624</td>
<td>no, error &gt;&gt;F60D</td>
</tr>
<tr>
<td>F629</td>
<td>check device status &lt;F55B&gt;</td>
</tr>
<tr>
<td>F629</td>
<td>error? &gt;&gt;F60D</td>
</tr>
<tr>
<td>F632</td>
<td>copy EOF from FCB (D815)</td>
</tr>
<tr>
<td>F640</td>
<td>copy caller's new EOF</td>
</tr>
<tr>
<td>F64B</td>
<td>compare old EOF to new (FE8A)</td>
</tr>
<tr>
<td>F651</td>
<td>if less than or equal to... &gt;&gt;F650</td>
</tr>
<tr>
<td>F653</td>
<td>if greater... &gt;&gt;F66D</td>
</tr>
<tr>
<td>F658</td>
<td>*** OLD EOF &lt;= NEW EOF ***</td>
</tr>
<tr>
<td>F66A</td>
<td>*** NO TRUNCATE NEEDED ***</td>
</tr>
<tr>
<td>F66B</td>
<td>new EOF beyond old</td>
</tr>
<tr>
<td>F66F</td>
<td>copy caller's EOF to FCB</td>
</tr>
<tr>
<td>F66A</td>
<td>exit by indicating flush needed &gt;&gt;FA66</td>
</tr>
<tr>
<td>F66D</td>
<td>*** OLD EOF &gt; NEW EOF ***</td>
</tr>
<tr>
<td>F67D</td>
<td>(** TRUNCATE FILE ***</td>
</tr>
<tr>
<td>F67D</td>
<td>flush first &lt;F526&gt;</td>
</tr>
<tr>
<td>F67D</td>
<td>error? &gt;&gt;F610</td>
</tr>
<tr>
<td>F672</td>
<td>$43/$49 --&gt; end of data block 1/0 buffer</td>
</tr>
<tr>
<td>F67C</td>
<td>compare current mark to new EOF (FE92)</td>
</tr>
<tr>
<td>F689</td>
<td>it is prior to EOF &gt;&gt;F6A2</td>
</tr>
<tr>
<td>F691</td>
<td>if past EOF, force mark back to EOF (FE92)</td>
</tr>
<tr>
<td>F6A2</td>
<td>construct EOF block number and (F689)</td>
</tr>
<tr>
<td>F6AA</td>
<td>byte offset into block from new (F6C6)</td>
</tr>
<tr>
<td>F6AB</td>
<td>EOF mark. (F6AB)</td>
</tr>
<tr>
<td>F6CA</td>
<td>on a block boundary? (F6C7)</td>
</tr>
<tr>
<td>F6C7</td>
<td>yes &gt;&gt;F6E2</td>
</tr>
<tr>
<td>F6CD</td>
<td>no, (F6C5)</td>
</tr>
<tr>
<td>F6C9</td>
<td>decrement block by 1</td>
</tr>
<tr>
<td>F6D7</td>
<td>but don't let it fall below 0</td>
</tr>
<tr>
<td>F6E2</td>
<td>copy key block number (FE92)</td>
</tr>
<tr>
<td>F6A2</td>
<td>set blocks freed to zero</td>
</tr>
<tr>
<td>F6BB</td>
<td>truncate file at new EOF &lt;FA78&gt;</td>
</tr>
<tr>
<td>F6BF</td>
<td>save status</td>
</tr>
<tr>
<td>F704</td>
<td>set new key block in FCB (FE8F)</td>
</tr>
<tr>
<td>F70A</td>
<td>drop FCB block count by number (D818)</td>
</tr>
<tr>
<td>F70D</td>
<td>of blocks freed in truncate routine. (F6C2)</td>
</tr>
</tbody>
</table>
ProDOS MLI -- V1.1.1 -- 18 SEP 84

F71A copy new storage type (FECL)
F727 turn off all block allocation flags <EDAP>
F72A update VCB free block count <F9F3>
F734 copy mark (D612)
F73C force current mark to infinity (D812)
F741 go set mark <EC48>
F746 no errors? >>F74F
F748 if error, indicate in saved status
F74E but continue
F74F copy caller's EOF to FCB <F656>
F752 flush and update <F526>
F755 no errors? >>F75E
F757 if error, indicate in saved status
F75D but continue
F75E ---
F760 exit

F761 *******************************************************
***** MLI GET EOF CALL *****
*******************************************************

F761 ---
F766 copy EOF to caller's list (Dw15)
F772 exit -- no errors

F773 *******************************************************
***** MLI NEW LINE CALL *****
*******************************************************

F773 ---
F775 copy newline mask
F77E and newline character
F784 return, no errors

F785 *******************************************************
***** MLI GET FILE INFO CALL *****
*******************************************************

F785 get the file entry <E5A3>
F786 ok? >>F7CC
F78A no, bad path?
F78D no, real error >>F789
F78F else, make it VOL DIR type
F791 with name length = 0 (FE5F)
F796 no free blocks needed (FE94)
F79C go through the motions to update the (FE91)
F79F VCB block count. <E97E>
F7A5 copy blocks free from VCB (D915)
F7B1 copy total blocks on volume to AUX_ID (D913)
F7BF total - free = blocks_used (FE94)
F7CC shift type down from high nibble (FE5F)

ProDOS MLI -- V1.1.1 -- 18 SEP 84

F7D8 copy the data to caller's paralist (FB0C)
F7E9 and exit

F7EA *******************************************************
***** MLI SET FILE INFO CALL *****
*******************************************************

F7EA get the file entry <E5A3>
F7ED error? >>F814
F7EF indicate backup needed now (BF95)
F7FE copy 13 params from caller's list to (FE0C)
F801 file entry staging area >>F808
F808 ---
F80D if any spurious access bits are on...
F811 access error!
F814 RETURN
F815 else, anything in his modification date?
F819 no >>F81E
F81B yes, go update directory >>E4C6
F81E ---
F81E no, use system date then update directory >>E4B6

F821 *******************************************************
***** MLI RENAME CALL *****
*******************************************************

F821 follow path to file <E5B6>
F824 ok? >>F863
F826 no, bad name?
F828 no, real error >>F842

*** RENAME VOLUME ***

F82A yes, copy new name <F94B>
F82D error? >>F842
F82F get first length (D700)
F833 get next (D700)
F836 bad path if more than one name for vol >>F8B7
F83B files open on volume? (D911)
F83E no, continue >>F844
F840 yes, file open error
F842 ---
F843 RETURN

F844 make type/len for a VOL DIR HDR
F84B write new name to VOL HDR <F93C>
F84E error? >>F899
F855 copy new name to device's VCB (D700)
F861 exit, no errors
F862 RETURN
 *** RENAME FILE ***

F863 get path index <F959>
F866 copy old name with prefix to my buffer (D70W)
F872 copy new name to buffer <F94B>
F875 error? >>F889
F877 get path index <F959>
F87D compare all levels of names up to and (DC0W)
F880 including the last. Find first which
F881 differ.
F885 save indices into names which point to (FEB9)
F888 final name. (FEBA)
F88B ---
F895 exit if they match completely
F896 RETURN

F897 index to differing new name (FEB9)
F89A point past it (D700)
F8A2 must be the last! (D700)
F8A5 it isn't >>F887
F8A7 it is, (FEBA)
F8AA do the same with the old name (DC0W)
F8B5 difference is only in last index? >>F8B8
F8B7 no, bad path error
F8B9 ---
F8BA RETURN

F8BB names good. follow path to new file <E5B6>
F8BE better get an error >>F8C4
F8CB if found, duplicate name in directory
F8C3 RETURN

F8C4 if error, better be file not found
F8C6 or else its really an error... >>F8B9
F8CB copy old pathname again <E0B9>
F8C8 get its file entry <E5A3>
F8CE error? >>F8B9
F8DD search FCB's <F833>
F8E0 exit if the file is open for write >>F8B9
F8DA does ACCESS permit rename?
F8DC yes >>F8E2
F8DE no, access error
F8ED ---
F8E1 RETURN

F8E2 get type/len from entry (FE5F)
F8E7 DIR file?
F8E9 yes, ok >>F8F3
F8EB seedling, sapling or tree?
F8ED yes, ok >>F8F3
Beneath Apple ProDOS Supplement

ProDOS MLI -- V1.1.1 -- 18 SEP 84  NEXT OBJECT ADDR: F989

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>F989</td>
<td>DESTROY enabled in ACCESS7 (FE7D)</td>
</tr>
<tr>
<td>F98E</td>
<td>yes &gt;&gt;F995</td>
</tr>
<tr>
<td>F990</td>
<td>no, access error</td>
</tr>
<tr>
<td>F995</td>
<td>check status of device (BF3W)</td>
</tr>
<tr>
<td>F998</td>
<td>error? &gt;&gt;F9B5</td>
</tr>
<tr>
<td>F99D</td>
<td>point to key block (FE7W)</td>
</tr>
<tr>
<td>F9AC</td>
<td>DIR file?</td>
</tr>
<tr>
<td>F9B0</td>
<td>no, &gt;&gt;F9B6</td>
</tr>
<tr>
<td>F9B2</td>
<td>yes, handle differently &gt;&gt;FA0E</td>
</tr>
<tr>
<td>F9B5</td>
<td>RETURN</td>
</tr>
</tbody>
</table>

*** DESTROY NON-DIRECTORY FILE ***

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>F9B6</td>
<td>set new storage type (FEC1)</td>
</tr>
<tr>
<td>F9B9</td>
<td>zero EOF mark (FEC1)</td>
</tr>
<tr>
<td>F9C3</td>
<td>byte offset = $200</td>
</tr>
<tr>
<td>F9CB</td>
<td>free all blocks in file &lt;FA78&gt;</td>
</tr>
<tr>
<td>F9CB</td>
<td>error? &gt;&gt;F9B5</td>
</tr>
<tr>
<td>F9CD</td>
<td>free key block of seedling (FEC8)</td>
</tr>
<tr>
<td>F9D6</td>
<td>error? &gt;&gt;F9B5</td>
</tr>
<tr>
<td>F9DB</td>
<td>mark DIR entry free</td>
</tr>
<tr>
<td>F9DD</td>
<td>decremenet DIR file count (FE5J)</td>
</tr>
<tr>
<td>F9E8</td>
<td>checkpoint volume bit map &lt;EB93&gt;</td>
</tr>
<tr>
<td>F9EB</td>
<td>error? &gt;&gt;F9B5</td>
</tr>
<tr>
<td>F9ED</td>
<td>update free block count in VCB &lt;F9F3&gt;</td>
</tr>
<tr>
<td>F9F0</td>
<td>and go update the directory &gt;&gt;E4B6</td>
</tr>
</tbody>
</table>

*** SUBROUTINE TO UPDATE FREE BLOCK ***

*** COUNT IN VCB ***

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>F9F3</td>
<td>add blocks freed to total free blocks (FE91)</td>
</tr>
<tr>
<td>F9F6</td>
<td>in VCB. (FEC2)</td>
</tr>
<tr>
<td>FA03</td>
<td>start next search for free blocks at</td>
</tr>
<tr>
<td>FA0A</td>
<td>start of bitmap. (D91C)</td>
</tr>
<tr>
<td>FA0D</td>
<td>exit</td>
</tr>
</tbody>
</table>

*** DESTROY DIRECTORY FILE ***

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>FA0E</td>
<td>DIR file?</td>
</tr>
<tr>
<td>FA10</td>
<td>no, error &gt;&gt;FA61</td>
</tr>
<tr>
<td>FA12</td>
<td>read volume bitmap block &lt;EB64&gt;</td>
</tr>
<tr>
<td>FA15</td>
<td>error? &gt;&gt;FA60</td>
</tr>
<tr>
<td>FA17</td>
<td>BLKNUM = key block pointer (FE7W)</td>
</tr>
<tr>
<td>FA21</td>
<td>read it &lt;EBEE&gt;</td>
</tr>
<tr>
<td>FA24</td>
<td>error? &gt;&gt;FA60</td>
</tr>
<tr>
<td>FA26</td>
<td>if DIR has any files... (DC25)</td>
</tr>
<tr>
<td>FA30</td>
<td>access error</td>
</tr>
<tr>
<td>FA35</td>
<td>write back block marking entry free (DCL4)</td>
</tr>
<tr>
<td>FA38</td>
<td>error? &gt;&gt;FA60</td>
</tr>
<tr>
<td>FA3D</td>
<td>if &quot;next_pointer&quot; is zero... (DC02)</td>
</tr>
</tbody>
</table>

ProDOS MLI -- V1.1.1 -- 18 SEP 84  NEXT OBJECT ADDR: FA47

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>FA47</td>
<td>go back and pretend it's a seedling &gt;&gt;F9CD</td>
</tr>
<tr>
<td>FA49</td>
<td>else, (DC03)</td>
</tr>
<tr>
<td>FA4C</td>
<td>free next block &lt;EA34&gt;</td>
</tr>
<tr>
<td>FA4F</td>
<td>error? &gt;&gt;FA60</td>
</tr>
<tr>
<td>FA51</td>
<td>BLKNUM = next block (DC02)</td>
</tr>
<tr>
<td>FA5B</td>
<td>read it &lt;EBEE&gt;</td>
</tr>
<tr>
<td>FA5E</td>
<td>if ok, continue in loop &gt;&gt;FA3D</td>
</tr>
<tr>
<td>FA66</td>
<td>else, error exit</td>
</tr>
<tr>
<td>FA61</td>
<td>incompatible file format error</td>
</tr>
<tr>
<td>FA66</td>
<td>******** SET WRITE OCCURRED FLAG ***********************</td>
</tr>
<tr>
<td>FA69</td>
<td>save some registers</td>
</tr>
<tr>
<td>FA74</td>
<td>indicate write occurred (FE92)</td>
</tr>
<tr>
<td>FA77</td>
<td>RETURN</td>
</tr>
<tr>
<td>FA78</td>
<td>******** TRUNCATE FILE AT EOF ********************</td>
</tr>
<tr>
<td>FA78</td>
<td>check storage type*16 (FEC1)</td>
</tr>
<tr>
<td>FA7B</td>
<td>seedling?</td>
</tr>
<tr>
<td>FA7D</td>
<td>yes &gt;&gt;FA8A</td>
</tr>
<tr>
<td>FA7F</td>
<td>no, sapling?</td>
</tr>
<tr>
<td>FA81</td>
<td>yes &gt;&gt;FA8D</td>
</tr>
<tr>
<td>FA83</td>
<td>no, tree?</td>
</tr>
<tr>
<td>FA85</td>
<td>yes &gt;&gt;FA90</td>
</tr>
<tr>
<td>FA87</td>
<td>no, die horribly &lt;BF8C&gt;</td>
</tr>
<tr>
<td>FA8A</td>
<td>go to seedling truncate &gt;&gt;F85C</td>
</tr>
<tr>
<td>FA8D</td>
<td>go to sapling truncate &gt;&gt;F823</td>
</tr>
<tr>
<td>FA90</td>
<td>truncate tree,</td>
</tr>
<tr>
<td>FA92</td>
<td>at most 128 blocks in master index (FEC8)</td>
</tr>
<tr>
<td>FA95</td>
<td>read the master index &lt;F887&gt;</td>
</tr>
<tr>
<td>FA96</td>
<td>error? &gt;&gt;FAF5</td>
</tr>
<tr>
<td>FA9A</td>
<td>at EOF yet? (FEC8)</td>
</tr>
<tr>
<td>FA90</td>
<td>yes &gt;&gt;FA6F</td>
</tr>
</tbody>
</table>

*** FREE WHOLE INDEX BLOCKS AFTER EOF ***

(free 8 subindex blocks each time the master index block is read since we must share its buffer)

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>FAA2</td>
<td>copy up to 8 non-zero index knob</td>
</tr>
<tr>
<td>FAA4</td>
<td>numbers to (DC00)</td>
</tr>
<tr>
<td>FAA7</td>
<td>a handy table (FECA)</td>
</tr>
<tr>
<td>FAA8</td>
<td></td>
</tr>
<tr>
<td>FAC1</td>
<td>if there weren't 8 left to do, zero (FEC8)</td>
</tr>
<tr>
<td>FAC4</td>
<td>remainder of the table (FED2)</td>
</tr>
<tr>
<td>FAC8</td>
<td></td>
</tr>
</tbody>
</table>
ProDOS MLI -- V1.1.1 -- 18 SEP 84

ADDR DESCRIPTION/CONTENTS

PAC8 update master index counter (FEC8)
FAD0 for all 8 entries: (FEC9)
FAD3 set BLKNUM (FEC9)
FA9B (exit when a 0 entry is found) >>FA95
FA9E read the sub-index block <EBEE>
FA9F error? >>FA95
FAB7 free all its blocks <FB96>
FA98 error? >>FA95
FAB0 and loop to do all 0 >>FAD0
FAB2 then go back and reread master index >>FA95
FA94 normal exit
FAB5 RETURN

FA6 now go free all the sub-index blocks (FEC4)
FA6A which follow EOF <FB98>
FA6D error? >>FA95
FA6F write back master index <EBEA>
FBB2 error? >>FA95
FBB4 EOF in first subindex? (FEC4)
FBB7 if so, demote to sapling file >>FBJ8
FBB9 else, BLKNUM < subindex block which (DC08)
FBB0 contains the EOF mark
FBB1 (exit if none there) >>FAA4
FBB8 else, read subindex block <EBEA>
FBBB and continue below >>FBB6
FB1D unless there is an error
FBAE demote tree to sapling <FB94>
FB21 error? >>FA95

*** TRUNCATE SAPLING FILE ***

FB23 read key block <FB97>
FB26 error? >>FA95
FB28 get LSB of block number (FEC5)
FB2C if zero, no blocks to free >>FB38
FB2E else, free rest of blocks in index <FB98>
FB31 following the EOF. check for error >>FA95
FB33 write index block back <EBEA>
FB36 error? >>FA95
FB38 get block number (FEC5)
FB3B might be block 0? >>FB52
FB3D no, get BLKNUM of data block (DC08)
FB44 from index block
FB45 (no block allocated?) >>FA94
FB4C read data block <EBEE>
FB4F and continue below >>FB61
FB51 unless error occurred

ProDOS MLI -- V1.1.1 -- 18 SEP 84

ADDR DESCRIPTION/CONTENTS

FB52 back to block 0? (FEC4)
FB55 no >>FB30
FB57 yes, demote to sapling <FB94>
FB5A error? >>FB96

*** TRUNCATE SAPLING FILE ***

FB5C read key block <FB97>
FB5F error? >>FB96
FB61 first page? (FEC7)
FB64 yes >>FB9C
FB67 no, better be second >>FB65
FB69 get byte offset (FEC6)
FB6C ---
FB6E zero beyond EOF mark (DC08)
FB7C in both pages if necessary (DC08)
FB82 then write block back and exit >>EBEA
FB85 exit normally
FB86 RETURN

FB87 ********** READ KEY BLOCK *****************************

FB87 BLKNUM = key block number (FEBF)
FB91 exit by reading the block >>EBEA

FB94 ********** DEMOTE FILE TO SMALLER FILE TYPE************

FB94 free block (FEC0)
FB9D error? >>FB95
FB9F get block from old index (DC08)
FBAC reduce storage type by one (FEC1)
FB94 and exit
FB95 RETURN

FBB6 ********** FREE ALL BLOCKS IN AN INDEX BLK *************

FBB6 ---
FBBA save BLKNUM
FBBE for each index entry after mark, (FEBD)
FB9 if it is non-zero....
FB90 free the block <EA34>
FB93 error? >>FB94
FB95 zero the index entry now (FEB9)
FB90 ---
FB91 loop through all entries >>FB9E
FB94 ---
FB96 restore old BLKNUM
FB9C and exit
**ProDOS MLI -- V1.1.1 -- 18 SEP 84**

**ADDR** DESCRIPTION/CONTENTS

---

**FBED** ********** ALLOCATE I/O BUFFER **********

**FBED** ---

**FBF** get I/O buffer page number
**FBF2** can't be below $800
**FBF4** else, error >>FC3B
**FBF6** can't be above $BC00
**FBF8** else, error >>FC3B
**FBFD** X4A/X4B --> I/O buffer
**FCB1** must be page aligned >>FC3B
**FC07** ---

**FC08** check each page of I/O buffer for <FC73>
**FC0B** prior allocation in system bit map (BF58)
**FC18** ---

**FC19** if ok, mark each page as allocated <FC73>
**FC1C** in system memory bit map (BF58)
**FC29** assign buffer number (REPNUM*2) in FCB (D600)
**FC31** and save buffer location in buffer list
**FC36** exit
**FC37** RETURN

**FC38** bad I/O buffer error
**FC3B** RETURN

**FC3C** ********** LOCATE I/O BUFFER **********

**FC3C** ---

**FC3D** AREG contains buffer number *2 (BF6E)
**FC40** move buffer pointer to NXTBUF variable (FEDD)
**FC49** exit

**FC4A** ********** FREE I/O BUFFER **********

**FC4A** is buffer already free? <FC3C>
**FC4D** yes, exit >>FC71
**FC53** zero its address in system global page (BF6F)
**FC60** ---

**FC61** free each page in buffer <FC73>
**FC64** by marking system bit map
**FC71** exit
**FC72** RETURN

**PC73** ********** LOCATE BIT MAP POSITION **********

(GIVEN PAGE NUMBER)

**PC73** XREG contains page number
**PC74** compute page number times 8
**PC77** use as offset for bitmask (P800)
**PC7E** page number / 8 = byte offset
**PC7F** into bitmap

---

**ProDOS MLI -- V1.1.1 -- 18 SEP 84**

**ADDR** DESCRIPTION/CONTENTS

---

**FCB1** exit

**FCB2** ********** CHECK BUFFER VALIDITY **********

START > $200 END < $BF00

**FCB2** get buffer address (MSB)
**FCB6** must be >$200 else error >>FC3B
**FCB8** get length (FEDB)
**FCBE** compute last page no. of buffer
**FC93** ---

**FC99** may not extend into $BF00
**FC9C** else, error >>FC3B

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

**FC9F** ---

**FCA0** see if this page is allocated <FC73>
**FCA6** if so, error >>FC3B
**FCA8** else, check other page also
**FCAE** then exit if both have been checked
**FCAD** RETURN

**FCAE** ********** MLI GET BUFF CALL **********

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

**FCBD** mark his buffer allocated
**FC2** error? >>FC4
**FCC4** get old buffer address (FEDD)
**FCGE** free old buffer's pages in map <FC59>
**FCDD** copy old buffer contents
**FC7D** to new buffer
**FCE3** then exit
**FCE4** RETURN

**FCE5** ********** GO TO QUIT CODE HANDLER **********

**FCE5** enable 2nd 4K bank of language card (C0B3)
**FCEB** (it lives at $D100-$D3FF) (C0B3)
**FC8B** Save zeropage $80 through $99 on stack
**FC77** Set ($80) --> $D100
**FC7F** Set ($82) --> $1000
ProDOS MLI -- V1.1.1 -- 18 SEP 84

ADDR  DESCRIPTION/CONTENTS

FD05  Set Y = Ø
FD06  3 pages of code to copy
FD08  ---
FD09  copy quit code handler to $1000
FD10  Restore zero page to original state
FD20  enable HIGH RAM BANK1 (C08B)
FD26  (MLI) (C0BB)
FD2B  point RSET vector at $1000 (03F2)
FD33  set power-up bytes properly
FD38  go to quit code handler at $1000 >>1000

FD3B  ********** NEW ROUTINE ********************
      THE ADDRESS OF THIS ROUTINE IS AT $36A.
      WE COULD NOT DETERMINE ITS PURPOSE.

FD3B  ---
FD3C  get current P-reg in accumulator
FD3D  save current P-reg
FD3E  clear overflow flag
FD3F  interrupts disabled?
FD41  no >>FD46
FD43  yes, set overflow flag (FD64)
FD46  disable interrupts
FD47  enable RAM.BANK2 (C083)
FD4D  set carry, indicating error
FD4E  pass a 5 to page 3 subroutine
FD50  call a page 3 subroutine <03D6>
FD53  store error number (BP0F)
FD56  enable RAM.BANK1 (C08B)
FD5C  restore original P-reg
FD5E  if error number is zero, (BP0F)
FD61  then indicate no error; >>FD64
FD63  otherwise indicate error
FD64  RETURN

FD65  **************** DATA AREA ********************

FD65  ********** MLI COMMAND TABLE ***************
      IN HASH CODE ORDER: IF COMMAND IS...
      ABCD EFGH (IN BINARY BITS)
      INDEX IS COMPUTED AS:
      Ø0Ø0 EFGH
      +Ø0ØØ ABCD

FD65  GET BUF
FD66  UNUSED
FD67  UNUSED
FD68  UNUSED
FD69  ALLOC INTERRUPT
### ProDOS MLI -- V1.1.1 -- 18 SEP 84

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>FD9A</td>
<td>NEWLINE</td>
</tr>
<tr>
<td>FD9B</td>
<td>READ</td>
</tr>
<tr>
<td>FD9C</td>
<td>WRITE</td>
</tr>
<tr>
<td>FD9D</td>
<td>CLOSE</td>
</tr>
<tr>
<td>FD9E</td>
<td>FLUSH</td>
</tr>
<tr>
<td>FD9F</td>
<td>SET MARK</td>
</tr>
<tr>
<td>FD00</td>
<td>GET MARK</td>
</tr>
<tr>
<td>FDA1</td>
<td>UNUSED</td>
</tr>
<tr>
<td>FDA2</td>
<td>SET EOF</td>
</tr>
<tr>
<td>FDA3</td>
<td>GET EOF</td>
</tr>
<tr>
<td>FDA4</td>
<td>SET BUF</td>
</tr>
<tr>
<td>FDA5</td>
<td></td>
</tr>
</tbody>
</table>

### MLI COMMAND ADDRESS TABLE

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>FDA5</td>
<td>CREATE</td>
</tr>
<tr>
<td>FDA7</td>
<td>DESTROY</td>
</tr>
<tr>
<td>FDA9</td>
<td>RENAME</td>
</tr>
<tr>
<td>FDB0</td>
<td>SET FILE INFO</td>
</tr>
<tr>
<td>FDB1</td>
<td>GET FILE INFO</td>
</tr>
<tr>
<td>FDB5</td>
<td>ON LINE</td>
</tr>
<tr>
<td>FDB6</td>
<td>SET PREFIX</td>
</tr>
<tr>
<td>FDB7</td>
<td>GET PREFIX</td>
</tr>
<tr>
<td>FDB8</td>
<td>OPEN</td>
</tr>
<tr>
<td>FDB9</td>
<td>NEWLINE</td>
</tr>
<tr>
<td>FDBB</td>
<td>READ</td>
</tr>
<tr>
<td>FDBD</td>
<td>WRITE</td>
</tr>
<tr>
<td>FDBF</td>
<td>CLOSE</td>
</tr>
<tr>
<td>FDC0</td>
<td>FLUSH</td>
</tr>
<tr>
<td>FDC1</td>
<td>SET MARK</td>
</tr>
<tr>
<td>FDC3</td>
<td>GET MARK</td>
</tr>
<tr>
<td>FDC5</td>
<td>SET EOF</td>
</tr>
<tr>
<td>FDC6</td>
<td>GET EOF</td>
</tr>
<tr>
<td>FDC7</td>
<td>SET BUF</td>
</tr>
<tr>
<td>FDC8</td>
<td>GET BUF</td>
</tr>
<tr>
<td>FDC9</td>
<td></td>
</tr>
<tr>
<td>FDCB</td>
<td></td>
</tr>
</tbody>
</table>

### MLI COMMAND INFO BYTE

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>FDCD</td>
<td></td>
</tr>
</tbody>
</table>

### CONSTANTS - DATA AREA

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>FDD6</td>
<td>0 1 0 - 09</td>
</tr>
<tr>
<td>FDD7</td>
<td>0 1 0 - 0A</td>
</tr>
<tr>
<td>FDD8</td>
<td>0 1 0 - 0B</td>
</tr>
<tr>
<td>FDD9</td>
<td>0 1 0 - 0C</td>
</tr>
<tr>
<td>FDDA</td>
<td>0 1 0 - 0D</td>
</tr>
<tr>
<td>FDDB</td>
<td>0 1 0 - 0E</td>
</tr>
<tr>
<td>FDDC</td>
<td>0 1 0 - 0F</td>
</tr>
<tr>
<td>FDDD</td>
<td>0 1 0 - 10</td>
</tr>
<tr>
<td>FDE0</td>
<td>0 1 0 - 11</td>
</tr>
<tr>
<td>FDE5</td>
<td>0 1 0 - 12</td>
</tr>
<tr>
<td>FDE6</td>
<td>0 1 0 - 13</td>
</tr>
</tbody>
</table>

---

**FDE1**

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>FDE1</td>
<td>Blocks Used</td>
</tr>
<tr>
<td>FDE3</td>
<td>End of File</td>
</tr>
<tr>
<td>FDE6</td>
<td>Special ID (Must be 5 bits on)</td>
</tr>
<tr>
<td>FDE7</td>
<td>'HUSTON:' Author's name</td>
</tr>
<tr>
<td>FDEE</td>
<td>Previous Block of Vol Dir Key Block</td>
</tr>
</tbody>
</table>

---

**FDF0**

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>FDF0</td>
<td>Version of ProDOS</td>
</tr>
<tr>
<td>FDF1</td>
<td>Minimum Version</td>
</tr>
<tr>
<td>FDF2</td>
<td>Access Byte (D</td>
</tr>
<tr>
<td>FDF3</td>
<td>Entry Length</td>
</tr>
<tr>
<td>FDF4</td>
<td>Entries per Block</td>
</tr>
<tr>
<td>FDF5</td>
<td>File Count</td>
</tr>
<tr>
<td>FDF6</td>
<td>Parent LSB (copied to SUBDIR HDR +$20)</td>
</tr>
</tbody>
</table>

---

**FDF8**

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>FDF8</td>
<td>File Type (Directory)</td>
</tr>
<tr>
<td>FDF9</td>
<td>Block Number</td>
</tr>
<tr>
<td>FDFB</td>
<td>Number of Blocks</td>
</tr>
<tr>
<td>FDFD</td>
<td>End of File</td>
</tr>
</tbody>
</table>

---

**FE00**

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>FE00</td>
<td>10000000</td>
</tr>
<tr>
<td>FE01</td>
<td>01000000</td>
</tr>
<tr>
<td>FE02</td>
<td>00100000</td>
</tr>
<tr>
<td>FE03</td>
<td>00010000</td>
</tr>
<tr>
<td>FE04</td>
<td>00001000</td>
</tr>
<tr>
<td>FE05</td>
<td>00000100</td>
</tr>
<tr>
<td>FE06</td>
<td>00000010</td>
</tr>
<tr>
<td>FE07</td>
<td>00000001</td>
</tr>
</tbody>
</table>

---

**FE00**

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>FE00</td>
<td>OFFSETS TO DATA AT $F300</td>
</tr>
</tbody>
</table>

---
PRODOS MLI -- V1.1.1 -- 18 SEP 84

ADDRESS DESCRIPTION/CONTENTS

FE00 Key Block
FE0A # Blocks Used
FE0C End of File

FE0F ********** SET/GET FILE_INFO OFFSETS **********************
FE0F Access
FE10 File Type
FE11 Aux Type
FE13 Storage Type
FE14 Blocks Used (MSB on means GET only no SET)
FE16 Datetime (Last Mod)
FE18 Datetime (Creation)

FE1E *********** FATAL ERROR MESSAGE *****************************
FE1E INSERT SYSTEM DISK AND RESTART
FE4F ---

FE4F ********** VARIABLES - DATA AREA *******************************

FE46 Parent Pointer Block
FE47 Parent Entry Number
FE48 Parent Entry Length
FE49 Datetime (Creation)
FE4E Version
FE4F Min Version
FE50 Access Byte
FE51 Entry Length
FE52 Entries per Block
FE53 File Count
FE55 Bit Map Pointer
FE57 Total Blocks
THE FOLLOWING 6 BYTES UNIQUELY IDENTIFY
A FILE:
FE59 Device Number
FE5A Current Directory Block Number (HDR)
FE5B Block Number of File Entry in Directory

PRODOS MLI -- V1.1.1 -- 18 SEP 84

ADDRESS DESCRIPTION/CONTENTS

FE5F ********** FILE ENTRY BUFFER ***********************
FE5F Type/Length (TTTTTTTT)
FE60 File Name (Max 15) >>3000
FE61 File Type
FE70 Key Pointer
FE72 Blocks Used
FE74 End of File
FE77 Datetime (Creation)
FE78 Version
FE7C Min Version
FE7D Access Attribute
FE7E Aux Type (Load Address/Record Length)
FE80 Datetime (Last Mod)
FE84 Header Pointer

FE86 ******** Variable Work Area **************************

FE86 3 Byte Scratch
FE89 ---
FE8A End of File
FE8D Previous Mark

FE8E Compare Vol Name Scratch
FE91 Offset into VCB Table ($D000)
FE92 Offset into FCB Table ($D080)
FE93 Free FCB found Flag

FE94 Number of Free Blocks needed
FE96 Storage Type
THE NUMBER OF ENTRIES EXAMINED OR...

FE97 FCB already open flag
FE98 File Count

FE99 Entries/Block Loop Count/Free FCB's refnum
Free Entry Found Flag (if > 0) or...
# of 1st bitmap block with free bit on or...

FE9B bit for free
FE9C # Blocks in Bitmap left to search
FE9D Y Register temp
FE9E Pathname Length

FE9F Devnum for Prefix Directory Header
FEA0 Block of Prefix Directory Header
FEA2 Bitmap Byte Offset in Page
FEA3 Bitmap Page Offset
FEA4 Bitmap Buffer Page (0 or 1)
Address Description/Contents

FEA5  Bitmap Flag (if $80, needs writing)
FEA6  Bitmap DEVNUM
FEA7  Bitmap Block Number
FEA9  Bitmap Block offset for Multiblock Bitmaps
       New Mark to be Positioned to or for Set Mark
       or New Moving Mark (for READ)
       or New EOF for SET_EOF
FEAD  Request Count (Read/Write etc.)
FEAF  Multi-Block I/O count
FEB0  Newline character
FEB1  Newline mask
FEB2  I/O Transfer occurred flag
FEB3  MLI Command * 2
FEB4  ORed into Access Flags ($20 - Backup)
FEB5  Duplicate Volume Flag (if $FF)
FEB6  Duplicate Volume's VCB index
FEB7  MLI function code (low 5 bits)
       Characters in current Pathname index lvl or
FEB8  ONLINE: volname len - loop index
FEB9  new pathname: index to last name
       old pathname: index to last name or..
FEBA  ONLINE: index to data buffer
FEBB  Old PFIXPTR value
FEBC  Pathname fully qualified flag (if $FF)
       Pathname: temp save area for index or..
FEBD  ONLINE: DEVCNT
FEBE  close-all error code
FEBF  Set EOF: new Key Block pointer
FEC1  New storage type (SET_EOF)
FEC2  Freed Blocks count
FEC4  EOF Block number (MSB then LSB)
FEC6  EOF byte offset into Block
FEC8  EOF - Master index counter
FEC9  Save area for index into table below
FECA  **************** DEVICE TABLE BUILT BY ONLINE *******************
       (also used by SET_EOF to keep track of
        8 blocks to be freed at a time)

FECA  device table part one
FED2  device table part two
FEDA  length of path, etc.
FEDD  next buffer address
FEDF  16 byte stack save area
FEFF  6 byte zero page save area
FEF5  Jump Vector, used for indirect jumps
**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.

### Jump Vectors
- BF00-BF02 **ENTRY**
  - JMP to MLI.
- BF03-BF05 **JSRARE**
  - JMP to system death code (via SBFPS).
- BF06-BF08 **DATETIME**
  - JMP to Date/Time routine (RTS if no clock).
- BF09-BF0B **SYSERR**
  - JMP to system error handler.
- BF0C-BF0E **SYSDEATH**
  - JMP to system death handler.
- BF0F **SERR**
  - System error number.

### Device Information
- **BF10-BF11** **DEVADR01**
  - Slot 0 reserved
- **BF12-BF13** **DEVADR11**
  - Slot 1, drive 1 device driver address.
- **BF14-BF15** **DEVADR21**
  - Slot 2, drive 1 device driver address.
- **BF16-BF17** **DEVADR31**
  - Slot 3, drive 1 device driver address.
- **BF18-BF19** **DEVADR41**
  - Slot 4, drive 1 device driver address.
- **BF1A-BF1B** **DEVADR51**
  - Slot 5, drive 1 device driver address.
- **BF1C-BF1D** **DEVADR61**
  - Slot 6, drive 1 device driver address.
- **BF1E-BF1F** **DEVADR71**
  - Slot 7, drive 1 device driver address.
- **BF20-BF21** **DEVADR02**
  - Slot 0 reserved.
- **BF22-BF23** **DEVADR12**
  - Slot 1, drive 2 device driver address.
- **BF24-BF25** **DEVADR22**
  - Slot 2, drive 2 device driver address.
- **BF26-BF27** **DEVADR32**
  - /RAM device driver address (need extra 64K).
- **BF28-BF29** **DEVADR42**
  - Slot 4, drive 2 device driver address.
- **BF2A-BF2B** **DEVADR52**
  - Slot 5, drive 2 device driver address.
- **BF2C-BF2D** **DEVADR62**
  - Slot 6, drive 2 device driver address.
- **BF2E-BF2F** **DEVADR72**
  - Slot 7, drive 2 device driver address.
- **BF30** **DEVNUM**
  - Slot and drive (DSSII0000) of last device.
- **BF31** **DEVCNT**
  - Count (minus 1) of active devices.
- **BF32-BF3F** **DEVLST**
  - List of active devices (slot, drive and identification—DSSIIII).

### Copyright Notice
- BF40-BF4F
- BF50-BF55 **IRQITX**
  - Switch in language card and call IRQ handler at $FFD8.
- BF56-BF57 **TEMP**
  - Temporary storage for IRQ code.
- BF58-BF5F **BITMAP**
  - Bitmap of low 48K of memory.
- BF70-BF71 **BUFFER1**
  - Open file 1 buffer address.
- BF72-BF73 **BUFFER2**
  - Open file 2 buffer address.
- BF74-BF75 **BUFFER3**
  - Open file 3 buffer address.
- BF76-BF77 **BUFFER4**
  - Open file 4 buffer address.
- BF78-BF79 **BUFFER5**
  - Open file 5 buffer address.
- BF7A-BF7B **BUFFER6**
  - Open file 6 buffer address.
- BF7C-BF7D **BUFFER7**
  - Open file 7 buffer address.
- BF7E-BF7F **BUFFER8**
  - Open file 8 buffer address.
### ProDOS System Global Page

<table>
<thead>
<tr>
<th>ADDR</th>
<th>LABEL</th>
<th>CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>BF80-BF81</td>
<td>INTERRUPT1</td>
<td>Interrupt handler address (highest priority).</td>
</tr>
<tr>
<td>BF82-BF83</td>
<td>INTERRUPT2</td>
<td>Interrupt handler address.</td>
</tr>
<tr>
<td>BF84-BF85</td>
<td>INTERRUPT3</td>
<td>Interrupt handler address.</td>
</tr>
<tr>
<td>BF86-BF87</td>
<td>INTERRUPT4</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>INTSPREG</td>
<td>S-register save area.</td>
</tr>
<tr>
<td>BF8C</td>
<td>INTFREG</td>
<td>P-register save area.</td>
</tr>
<tr>
<td>BF8D</td>
<td>INTBIKID</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>YYYYMMYY MMMMM.DDDD.</td>
</tr>
<tr>
<td>BF92-BF93</td>
<td>TIME</td>
<td>...HHHHH...MMHHH.</td>
</tr>
<tr>
<td>BF94</td>
<td>LEVEL</td>
<td>Current file level.</td>
</tr>
<tr>
<td>BF95</td>
<td>BUBIT</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>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ADDR</th>
<th>LABEL</th>
<th>CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>BF99</td>
<td>SLTBYTE</td>
<td>0...0...II</td>
</tr>
<tr>
<td>BF9A</td>
<td>PFIIXPTR</td>
<td>0...0...II+</td>
</tr>
<tr>
<td>BF9B</td>
<td>MLIACTV</td>
<td>0...0...Ile</td>
</tr>
<tr>
<td>BF9C-BF9D</td>
<td>CMDADDR</td>
<td>IIII emulation</td>
</tr>
<tr>
<td>BF9E</td>
<td>SAVEX</td>
<td>00...1...Future expansion</td>
</tr>
<tr>
<td>BF9F</td>
<td>SAVEY</td>
<td>01...1...Future expansion</td>
</tr>
</tbody>
</table>

### ProDOS System Global Page

<table>
<thead>
<tr>
<th>ADDR</th>
<th>LABEL</th>
<th>CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>BFA0-BFCF</td>
<td>Language Card Bank Switching Routines</td>
<td>Language card entry and exit routines.</td>
</tr>
<tr>
<td>BFA0</td>
<td>EXIT</td>
<td></td>
</tr>
<tr>
<td>BFA1</td>
<td>EXIT1</td>
<td></td>
</tr>
<tr>
<td>BFA2</td>
<td>EXIT2</td>
<td></td>
</tr>
<tr>
<td>BFB7</td>
<td>MLIENT1</td>
<td></td>
</tr>
<tr>
<td>BFD0-BFF3</td>
<td>Interrupt Routines</td>
<td>Interrupt entry and exit routines.</td>
</tr>
<tr>
<td>BFD0</td>
<td>IRQXIT</td>
<td></td>
</tr>
<tr>
<td>BFD1</td>
<td>IRQXIT1</td>
<td></td>
</tr>
<tr>
<td>BFD2</td>
<td>IRQXIT2</td>
<td></td>
</tr>
<tr>
<td>BFE7</td>
<td>ROMXIT</td>
<td></td>
</tr>
<tr>
<td>BFE8</td>
<td>IRQENT</td>
<td></td>
</tr>
<tr>
<td>BFF4</td>
<td>BNKBYT1</td>
<td>Storage for byte at $00.</td>
</tr>
<tr>
<td>BFF5</td>
<td>BNKBYT2</td>
<td>Storage for byte at $80.</td>
</tr>
<tr>
<td>BFF6-BFFB</td>
<td>Switch on language card and call system death handler ($DF1).</td>
<td></td>
</tr>
<tr>
<td>BFFC</td>
<td>IBAKVER</td>
<td>Minimum version of Kernel needed for this interpreter.</td>
</tr>
<tr>
<td>BFFD</td>
<td>IVERSION</td>
<td>Version number of this interpreter.</td>
</tr>
<tr>
<td>BFFE</td>
<td>KBAKVER</td>
<td>Minimum version of Kernel compatible with this Kernel.</td>
</tr>
<tr>
<td>BFFF</td>
<td>KVERSION</td>
<td>Version number of this Kernel.</td>
</tr>
</tbody>
</table>
Beneath Apple ProDOS Supplement

ProDOS QUIT Code -- V1.1.1 -- 18 SEP 84 NEXT OBJECT ADDR: 1000

ADDR DESCRIPTION/CONTENTS

1000 MODULE STARTING ADDRESS

*******************************************************************************
* QUIT Code
* Stored in BANK2 of High RAM
* at $D100 and moved to $1000
* by an MLI routine at $FC35,
* which JMPS to $1000.
* VERSION 1.1.1 -- 18 SEP 84
* (The QUIT code is still the same
* as it was in Version 1.0.1)
*******************************************************************************

1000 ********* INITIALIZATION **************

1000 Select ROM (C082)
1003 Set Video (FE93)
1006 Set Keyboard (FE99)
1009 Disable 80 column card (C00C)
100C Select Alternate character set (C00F)
100F Disable 80 column store (C000)

1012 ********** INITIALIZE MEMORY BITMAP ***************

1012 Mark pages $0, $1, $4 through $7
1014 and $8F as in use

1027 ********** DISPLAY CURRENT PREFIX ***************

1027 Clear Screen and Home cursor (FC58)
102A Go down 1 line (FDB8)
102D Get Pointer to Prompt1 (Prefix)
102F and store it in Print Routine (11E9)
1037 Call Print Routine (11E6)
103A Position to line 3
1041 Call MLI (GET_PREFIX) (BF00)
1044 Data: GET_PREFIX command number
1045 Data: Pointer to Parameter list
1047 Terminate Prefix with 0 (F200)
104A for Print routine
104F Get Pointer to Prefix
1051 and store it in Print Routine (11E9)
1059 And Print it (11E6)

105C ********** GET PREFIX NAME ***************

105C Initialize counter
1063 Read a key (FDC9)
1066 Is it CARRIAGE RETURN?
1068 Yes, then accept Prefix >>108B
106A No, then save character
106B Clear to end of line (FC9C)
106E Retrieve character
106F Is it ESCAPE?
1071 Yes, then start all over again >>1027
1073 Is it CANCEL?
1075 Yes, then start all over again >>1027
1077 Is it TAB?
1079 Yes, then sound Bell, get another character >>108E
107B Is it BACKSPACE?
107D No, then keep checking >>108C
107F Yes, then is there room to move back?
1081 No, then don’t try >>1086

1000 ********* ZERO PAGE EQUATES **************

0024 Cursor Horizontal
0025 Cursor Vertical

1000 ********** EXTERNAL EQUATES **************

0280 Prefix Buffer
1000 Buffer
2000 Buffer
BF00 MLI Entry
BF58 Bitmap

1000 ********** SOFT SWITCHES **************

C000 Keyboard
C008 Disable 80 column store
C00C Disable 80 column card
C00F Select alternate character set
C010 Keyboard Strobe
C022 ROM select

1000 ********** MONITOR EQUATES **************

FC58 Home
FC9C Clear to end of line
FDBC Read a key
FD8E Output a Carriage Return
FDDE Output a Character
FE89 Set Keyboard
FF93 Set Video
FF3A Sound Bell
ProDOS QUIT Code -- VI.1.1 -- 18 SEP 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>1085</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;FF3A&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;,&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;,&quot;,&quot;?</td>
</tr>
<tr>
<td>109A</td>
<td>Yes, Invalid - try again &gt;&gt;108E</td>
</tr>
<tr>
<td>1092</td>
<td>Is it less than or equal to &quot;9&quot;?</td>
</tr>
<tr>
<td>109A</td>
<td>Yes, keep checking &gt;&gt;109A</td>
</tr>
<tr>
<td>109A</td>
<td>Is it less than &quot;,&quot;,&quot;?</td>
</tr>
<tr>
<td>109B</td>
<td>Yes, Invalid - try again &gt;&gt;108E</td>
</tr>
<tr>
<td>109A</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;FDE0&gt;</td>
</tr>
<tr>
<td>10B5</td>
<td>Go back 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;FF3A&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>
<tr>
<td>10CE</td>
<td>Clear Screen and Home cursor &lt;FC98&gt;</td>
</tr>
<tr>
<td>10D1</td>
<td>Go down 1 line &lt;FDE0&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>10E2</td>
<td>Initialize counter</td>
</tr>
<tr>
<td>10FA</td>
<td>Output a SUB</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 (C810)</td>
</tr>
</tbody>
</table>

ProDOS QUIT Code -- VI.1.1 -- 18 SEP 84

<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;1103</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;11D0</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;FF3A&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;Z&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;,&quot;?</td>
</tr>
<tr>
<td>1126</td>
<td>Valid character - try again &gt;&gt;1114</td>
</tr>
<tr>
<td>1128</td>
<td>Is it greater than &quot;,&quot;,&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;,&quot;,&quot;?</td>
</tr>
<tr>
<td>112E</td>
<td>Yes, keep checking &gt;&gt;1114</td>
</tr>
<tr>
<td>1130</td>
<td>Is it less than &quot;,&quot;,&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>113B</td>
<td>Retrieve character</td>
</tr>
<tr>
<td>113D</td>
<td>Print it &lt;FDE0&gt;</td>
</tr>
<tr>
<td>113F</td>
<td>Increment counter</td>
</tr>
<tr>
<td>113D</td>
<td>Sound 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>
<tr>
<td>1147</td>
<td>Load and EXECUTE APPLICATION</td>
</tr>
</tbody>
</table>

1147 | Output a blank |
| 114C | Store length of Application name (0280) |
| 114F | Call MLI (GET FILE_INFO) <BF00> |
| 1152 | Data: GET_FILE_INFO command number |
| 1153 | Data: Pointer to Parameter list |
| 1155 | Continue if no error >>115A |
| 1157 | Else, go to Error Handler >>11F6 |
ProDOS QUIT Code -- V1.1.1 -- 18 SEP 84

ADDRE DESCRIPTION/CONTENTS

115A Get File Type (12D5)
115B Is it ProDOS System file?
115C Yes, continue >>1166
115D No, indicate Error $01
115E Go to Error Handler >>11F6

1166 Set Reference number to 0
1167 Call MLI (CLOSE) <BP00>
1168 Data: CLOSE command number
1169 Data: Pointer to Parameter list
1170 Continue if no error >>1176
1171 Else, go to Error Handler >>11F6
1172 Get Access Byte (12D4)
1173 Yes, >>1182
1174 No, Indicate Error $27
1175 Go to Error Handler >>11F6

1182 Call MLI (OPEN) <BP00>
1183 Data: OPEN command number
1184 Data: Pointer to Parameter list
1185 Continue if no error >>118D
1186 Else, go to Error Handler >>11F6

118D Get Reference Number (12B8)
118E and update READ and (12EC)
118F GET_EOF parameter lists (12F4)
1190 Call MLI (GET_EOF) <BP00>
1191 Data: GET_EOF command number
1192 Data: Pointer to Parameter list
1193 Continue if no error >>11A1
1194 Else, go to Error Handler >>11F6

11A1 Is EOF mark less than $10000 (12F7)
11A2 Yes, continue on >>11AB
11A3 No, Indicate Error $27
11A4 Go to Error Handler >>11F6

11AB Transfer EOF to Request count (12F5)
11AC in READ parameter list (12EF)
11AE Call MLI (READ) <BP00>
11BB Data: READ command number
11BD Save status of READ
11BE Call MLI (CLOSE) <BP00>
11CD Data: Get Prefix command number
11CE Data: Pointer to Parameter list
11CF Continue if no error >>11CA
11C0 Else, retrieve status
11C1 and go to Error Handler >>11F6

11CA Was READ good?
11CB No, go to Error Handler >>11F7
11CD Yes, execute application >>2000

11D0 Backspace ROUTINE

11D0 Get cursor position horizontal
11D2 If 0 exit routine >>11E3
11D4 Decrement counter
11D5 Output a space
11D7 Output cursor back 2 spaces
11D8 Output a space <PED4>
11E0 Move cursor back 1 space
11E1 Return to get another character >>10EA

11E6 PRINT TEXT ROUTINE

11E6 Initialize offset
11E8 Get a character (11E8)
11EB If it is 0 then exit >>11F5
11EF Output it <PE0>2
11F2 Increment offset
11F3 Get another character unless we've done 256 >>11E8
11F5 Return to caller

11F6 PRINT ERROR MESSAGE

11F6 Save Accumulator (Error Number)
11F81c Get Error number
1201 Is it $01?
1203 No, then keep checking >>1211
1205 Get Pointer to Error1 (No System file)
1207 and store it in Print Routine (11E9)
120F Branch always taken >>1237
1211 Is it $40?
1213 Yes, then indicate Error3 >>122D
1215 Is it $44?
1217 Yes, then indicate Error3 >>122D
1219 Is it $45?
121B Yes, then indicate Error3 >>122D
121D Is it $46?
121F Yes, then indicate Error3 >>122D
1221 Else, Get Pointer to Error2 (1/O Error)
1223 and go back in Print Routine (11E9)
122B Branch always taken >>1237
122D Get Pointer to Error3 (Path not found)
122F and store it in Print Routine (11E9)
1237 Print Error message <11E6>
123A Position to line 0
ProDOS QUIT Code -- V1.1.1 -- 18 SEP 84

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'

120D1 ********** 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

12E2 READ Parmlist
  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
  12F8 Parmcount
  12F9 Pathname

12FB ********** $12FB-$12FF UNUSED ***********************

12FB These unused bytes are $D3FB-$D3FF in high RAM
12FF and $59FB-$59FF when loaded as part of "PRODOS" file.
Disk II Device Driver -- V1.1.1 -- 18 SEP 84  NEXT OBJECT ADDR: D000

ADDR DESCRIPTION/CONTENTS

D000 MODULE STARTING ADDRESS

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

* 5.25" DISK DEVICE DRIVER *
* RESIDES AT $D000-$D0FF *
* VERSION 1.1.1 -- 18 SEP 84 *
******************************************************************************

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

003A Checksum
003B Workbyte
003C 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)

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

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

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

C080 Phase Zero Off
C088 Motor Off
C089 Motor On
C09A Drive Select
C09C Read Data Register
C09D Write Data Register
C09E Set Read Mode
C09F Set Write Mode
C09E Read Data Register (slot 6)

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

D000 Clear decimal mode
D001 Clear phases in case IWM device in this slot <D6BE>
D004 Five NOP's so code below will
D005 fit up against Table at $D196
D009 Check validity of calling parameters <D6D0>
D00C If not valid exit with error >>D034
D00E Convert Block Number to a Track and Sector

Disk II Device Driver -- V1.1.1 -- 18 SEP 84  NEXT OBJECT ADDR: D010

ADDR DESCRIPTION/CONTENTS

D010 ----
D014 0000000T TTTTABC
D015 . . >>D010
D017 . .
D018 . . >>D01C
D01A 00TTTTT 0000C0A
D01C ----
D020 Preserve Sector Number
D021 Execute command <D038>
D024 Restore Sector Number - was prior action ok?
D025 No, then exit >>D030
D027 Increment Buffer Pointer
D029 Increment Sector Number by 2 for rest of block
D02B Execute command <D039>
D02E Decrement Buffer Pointer (to start of block
D030 Get error number (if any - 0 indicates no error) (D358)
D033 Return to caller

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

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

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

D03B Set recalibration count to 1
D03D Preserve sector number (D357)
D040 Get "Unitnum" DSSS000
D042 Strip out Drive DSSS000
D043 Preserve slot number
D046 Check for slot change, turn off motor if so <D69B>
D049 See if motor is on <D4DA>
D04C Save test results
D04F Initialize counter for delay routine (D370)
D054 See if slot or drive has changed (D359)
D057 Update "current" unit number (D359)
D05A Save test results
D05B Put drive number in Carry flag
D05C Turn motor on (C089)
D062 Select appropriate drive (C08A)
D065 Check test results - Same slot/drive?
D066 Yes, then skip delay >>D072
D069 Wait for new Drive
D06B to come up to speed <D385>
D072 Is command a status request?
D074 Yes, then do not move disk arm >>D07C
D076 Get track number for current request (D356)
D079 And go there <D10C>
D07D Check test results - Was motor on?
D07D Yes, then skip delay >>D08E
<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>D07F</td>
<td>Wait for Drive to</td>
</tr>
<tr>
<td>D081</td>
<td>come up to speed &lt;D385&gt;</td>
</tr>
<tr>
<td>D089</td>
<td>Is motor on yet? &lt;D4DA&gt;</td>
</tr>
<tr>
<td>D08C</td>
<td>No, then exit with error &gt;&gt;D0EA</td>
</tr>
<tr>
<td>D08E</td>
<td>Is command a &quot;status&quot; request?</td>
</tr>
<tr>
<td>D090</td>
<td>Yes, then determine status &gt;&gt;D0FD</td>
</tr>
<tr>
<td>D092</td>
<td>Is command a &quot;read&quot; request?</td>
</tr>
<tr>
<td>D093</td>
<td>Yes, then continue on &gt;&gt;D098</td>
</tr>
<tr>
<td>D095</td>
<td>Prepare data for write (prenibblize) &lt;D5F0&gt;</td>
</tr>
<tr>
<td>D09A</td>
<td>---</td>
</tr>
<tr>
<td>D09A</td>
<td>Initialize &quot;retry&quot; count at 64 (D369)</td>
</tr>
<tr>
<td>D09D</td>
<td>---</td>
</tr>
<tr>
<td>D09F</td>
<td>Read an address field - Good read? &lt;D398&gt;</td>
</tr>
<tr>
<td>D0A2</td>
<td>Yes, then continue on &gt;&gt;D0BE</td>
</tr>
<tr>
<td>D0A4</td>
<td>Decrement &quot;retry&quot; count - More to try? (D369)</td>
</tr>
<tr>
<td>D0A7</td>
<td>Yes, then try again &gt;&gt;D09D</td>
</tr>
<tr>
<td>D0A9</td>
<td>No, just in case indicate &quot;I/O Error&quot;</td>
</tr>
<tr>
<td>D0AB</td>
<td>Decrement &quot;recalibration&quot; count - More to try? (D36A)</td>
</tr>
<tr>
<td>D0AE</td>
<td>No, then exit with error &gt;&gt;D0EA</td>
</tr>
<tr>
<td>D0B0</td>
<td>Get &quot;current&quot; track (D35A)</td>
</tr>
<tr>
<td>D0B3</td>
<td>Preserve it</td>
</tr>
<tr>
<td>D0B4</td>
<td>Double it and</td>
</tr>
<tr>
<td>D0B5</td>
<td>add 16 to it for recalibration</td>
</tr>
<tr>
<td>D0B7</td>
<td>Reinitialize Retry Count</td>
</tr>
<tr>
<td>D0BC</td>
<td>Branch always taken &gt;&gt;D0CC</td>
</tr>
<tr>
<td>D0C1</td>
<td>Was the right track found? (D35A)</td>
</tr>
<tr>
<td>D0C4</td>
<td>Yes, then continue on &gt;&gt;D0D5</td>
</tr>
<tr>
<td>D0C6</td>
<td>Get &quot;current&quot; track (D35A)</td>
</tr>
<tr>
<td>D0C9</td>
<td>Preserve it</td>
</tr>
<tr>
<td>D0CA</td>
<td>Get track we found</td>
</tr>
<tr>
<td>D0CB</td>
<td>Double it</td>
</tr>
<tr>
<td>D0CC</td>
<td>Put new value in Device Track Table &lt;D4D3&gt;</td>
</tr>
<tr>
<td>D0CF</td>
<td>Get track we want</td>
</tr>
<tr>
<td>D0D0</td>
<td>And go there &lt;D18C&gt;</td>
</tr>
<tr>
<td>D0D3</td>
<td>Branch always taken &gt;&gt;D09D</td>
</tr>
<tr>
<td>D0DB</td>
<td>Was the right sector found? (D357)</td>
</tr>
<tr>
<td>D0DF</td>
<td>No, then try again &gt;&gt;D0A4</td>
</tr>
<tr>
<td>D0F0</td>
<td>Is command a &quot;write&quot; request?</td>
</tr>
<tr>
<td>D0F2</td>
<td>Yes, then go do it &gt;&gt;D0F4</td>
</tr>
<tr>
<td>D0F2</td>
<td>Read the data - Good read? &lt;D3F0&gt;</td>
</tr>
<tr>
<td>D0F5</td>
<td>No, then try again &gt;&gt;D0A4</td>
</tr>
<tr>
<td>D0F7</td>
<td>Indicate no errors</td>
</tr>
<tr>
<td>D0F9</td>
<td>BNE Instruction, never taken</td>
</tr>
<tr>
<td>D0FA</td>
<td>Indicate error</td>
</tr>
<tr>
<td>D0FB</td>
<td>Preserve error number (D358)</td>
</tr>
<tr>
<td>D0FE</td>
<td>Get Slot</td>
</tr>
<tr>
<td>D0F0</td>
<td>Turn motor off (C068)</td>
</tr>
<tr>
<td>D0F3</td>
<td>Return to caller</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>D0F3</td>
<td>HANDLE WRITE REQUEST</td>
</tr>
<tr>
<td>D0F4</td>
<td>Write data - Good write? &lt;D508&gt;</td>
</tr>
<tr>
<td>D0F7</td>
<td>Yes, then exit &gt;&gt;D0E7</td>
</tr>
<tr>
<td>D0F9</td>
<td>Indicate &quot;Write-protect error&quot;</td>
</tr>
<tr>
<td>D0FB</td>
<td>Branch always taken &gt;&gt;D0EA</td>
</tr>
<tr>
<td>D0FD</td>
<td>GET STATUS</td>
</tr>
<tr>
<td>D0FD</td>
<td>Get Slot number</td>
</tr>
<tr>
<td>D102</td>
<td>Check &quot;write-protect&quot; status (C0BE)</td>
</tr>
<tr>
<td>D106</td>
<td>Put result in Carry flag</td>
</tr>
<tr>
<td>D106</td>
<td>Select read mode (C06C)</td>
</tr>
<tr>
<td>D109</td>
<td>Exit with appropriate status &gt;&gt;D0F7</td>
</tr>
<tr>
<td>D10C</td>
<td>LOCATE DESIRED TRACK</td>
</tr>
<tr>
<td>D10C</td>
<td>Double the track number for proper phase</td>
</tr>
<tr>
<td>D10D</td>
<td>Preserve destination track * 2 (D36F)</td>
</tr>
<tr>
<td>D110</td>
<td>Turn all phases off &lt;D125&gt;</td>
</tr>
<tr>
<td>D113</td>
<td>Get offset into Device Track Table &lt;D4F1&gt;</td>
</tr>
<tr>
<td>D116</td>
<td>Get track (D359)</td>
</tr>
<tr>
<td>D119</td>
<td>Update &quot;current&quot; track (D35A)</td>
</tr>
<tr>
<td>D121</td>
<td>Get destination track (D36F)</td>
</tr>
<tr>
<td>D121</td>
<td>Get track from Device Track Table (D359)</td>
</tr>
<tr>
<td>D122</td>
<td>Move arm to desired track &lt;D133&gt;</td>
</tr>
<tr>
<td>D125</td>
<td>Initialize phase number, starting with 3</td>
</tr>
<tr>
<td>D127</td>
<td>---</td>
</tr>
<tr>
<td>D128</td>
<td>Decrement phase number - More to do?</td>
</tr>
<tr>
<td>D12C</td>
<td>Yes, then continue until all phases done &gt;&gt;D127</td>
</tr>
<tr>
<td>D12E</td>
<td>Divide track number by 2 (D35A)</td>
</tr>
<tr>
<td>D132</td>
<td>Return to caller</td>
</tr>
<tr>
<td>D133</td>
<td>ARM MOVE ROUTINE</td>
</tr>
<tr>
<td>D133</td>
<td>Preserve track to find (D372)</td>
</tr>
<tr>
<td>D136</td>
<td>Are we already there? (D35A)</td>
</tr>
<tr>
<td>D139</td>
<td>Yes, then set appropriate phase and exit &gt;&gt;D137</td>
</tr>
<tr>
<td>D13D</td>
<td>Initialize phase count (halftracks) (D368)</td>
</tr>
<tr>
<td>D143</td>
<td>Preserve &quot;current&quot; track for comparisons (D371)</td>
</tr>
<tr>
<td>D146</td>
<td>Subtract track to find to compute delta-tracks</td>
</tr>
<tr>
<td>D147</td>
<td>Are we already there? (D372)</td>
</tr>
<tr>
<td>D14A</td>
<td>Yes, then clear prior phase and exit &gt;&gt;D133</td>
</tr>
<tr>
<td>D14C</td>
<td>Positive delta-tracks - go move arm out &gt;&gt;D155</td>
</tr>
<tr>
<td>D14E</td>
<td>Negative delta-tracks - Get absolute value delta-tracks less 1</td>
</tr>
<tr>
<td>D150</td>
<td>Increment current phase to move in (D35A)</td>
</tr>
<tr>
<td>D153</td>
<td>Branch always taken &gt;&gt;D13A</td>
</tr>
<tr>
<td>D155</td>
<td>Compute absolute value delta-tracks less 1</td>
</tr>
<tr>
<td>D157</td>
<td>Decrement current phase to move out (D35A)</td>
</tr>
<tr>
<td>Disk II Device Driver -- V1.1.1 -- 18 SEP 84 NEXT OBJECT ADDR: D15A</td>
<td></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>D15A</td>
<td>Compare delta-tracks with phases moved (D36B)</td>
</tr>
<tr>
<td>D15D</td>
<td>Use smaller value for offset to delay tables &gt;&gt;D162</td>
</tr>
<tr>
<td>D162</td>
<td>Are we pointing at last table value yet?</td>
</tr>
<tr>
<td>D164</td>
<td>Yes, then continue to use current offset &gt;&gt;D168</td>
</tr>
<tr>
<td>D166</td>
<td>Else, use new offset</td>
</tr>
<tr>
<td>D167</td>
<td>Set Carry flag for set phase operation</td>
</tr>
<tr>
<td>D168</td>
<td>Set a phase &lt;D187&gt;</td>
</tr>
<tr>
<td>D168</td>
<td>Get delay value from table (D373)</td>
</tr>
<tr>
<td>D16E</td>
<td>Delay &lt;D385&gt;</td>
</tr>
<tr>
<td>D171</td>
<td>Get prior phase number (D371)</td>
</tr>
<tr>
<td>D174</td>
<td>Clear Carry flag for clear phase operation</td>
</tr>
<tr>
<td>D175</td>
<td>Clear a phase &lt;D18A&gt;</td>
</tr>
<tr>
<td>D178</td>
<td>Get delay value from table (D37C)</td>
</tr>
<tr>
<td>D17B</td>
<td>Delay &lt;D385&gt;</td>
</tr>
<tr>
<td>D17E</td>
<td>Increment phases moved (D36B)</td>
</tr>
<tr>
<td>D183</td>
<td>Delay &lt;D385&gt;</td>
</tr>
<tr>
<td>D187</td>
<td>Get &quot;current&quot; phase number (D35A)</td>
</tr>
<tr>
<td>D18A</td>
<td>Use low two bits only, zero to three - 00000000</td>
</tr>
<tr>
<td>D18C</td>
<td>Multiply by two and bring in Carry - 00000000</td>
</tr>
<tr>
<td>D18D</td>
<td>Merge in slot number - 00000000</td>
</tr>
<tr>
<td>D18F</td>
<td>Put in X-reg for following operation</td>
</tr>
<tr>
<td>D190</td>
<td>Toggle appropriate phase (C080)</td>
</tr>
<tr>
<td>D193</td>
<td>Restore slot number to X-reg</td>
</tr>
<tr>
<td>D195</td>
<td>Return to caller</td>
</tr>
</tbody>
</table>

**TABLE 1**

<table>
<thead>
<tr>
<th>Read Translate Table with PreNiblize</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit mask Tables and Epilog Table in unused areas</td>
</tr>
</tbody>
</table>

**TABLE 2**

<table>
<thead>
<tr>
<th>Read Translate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit Mask 1</td>
</tr>
<tr>
<td>Bit Mask 2</td>
</tr>
</tbody>
</table>

**Disk Device Track Table**

<table>
<thead>
<tr>
<th>Table Entry</th>
</tr>
</thead>
<tbody>
<tr>
<td>Track number</td>
</tr>
<tr>
<td>Sector number</td>
</tr>
<tr>
<td>Error number</td>
</tr>
</tbody>
</table>

| Slot 1, Devices 1 & 2 |
| Slot 2, Devices 1 & 2 |
| Slot 3, Devices 1 & 2 |
| Slot 4, Devices 1 & 2 |
| Slot 5, Devices 1 & 2 |
| Slot 6, Devices 1 & 2 |
| Slot 7, Devices 1 & 2 |
Disk II Device Driver -- V1.1.1 -- 18 SEP 84  NEXT OBJECT ADDR: D367

ADDR  DESCRIPTION/CONTENTS

D369  Retry count (initially 64)
D36A  Recalibration count (initially 1)
D36B  Counter for Read Address routine
D36C  Temporary storage for Read Address routine
D36D  Track counter for Arm Move routine
D36E  Checksum computation
D36F  Volume found
D370  Sector found
D371  Delay counter (low byte)
D372  Track found
D373  Delay counter (high byte)
D374  Prior Track
D375  Track number for Arm Move routine

D376 ********** PHASEON/PHASEOFF TABLES ***************

D377 Phase on table (delays for disk head acceleration)
D378 Phase off table (delays for disk head deceleration)

D379 ********** WAIT ROUTINE ***********************

D380 Wait about 100 times A-register (microseconds)
D381 ---
D382 ---
D383 Return to caller

D384 ********** READ ADDRESS FIELD ***********************

D385 Initialize "must find" count to $FC
D386 Increment count (low order byte) - Zero yet?
D387 No, skip ahead >>D3A5
D388 Increment count (high order byte) - Zero yet? (D36B)
D389 Yes, exit and indicate Read Error >>D3FB
D38A Read data register (C08C)
D38B Loop until data valid >>D3A5
D38C Is it first address mark ($D5)?
D38D No, then increment "must find" count >>D39D
D38E Delay for data latch to clear
D38F Read data register (C08C)
D390 Loop until data valid >>D3AF
D391 Is it second address mark ($AA)?
D392 No, then see if it's first address mark >>D3AA
D393 Initialize count for four byte read
D394 Read data register (C08C)
D395 Loop until data valid >>D3BA
D396 Is it third address mark ($96)?
D397 No, then see if it's first address mark >>D3AA
D398 Set interrupt flag

Disk II Device Driver -- V1.1.1 -- 18 SEP 84  NEXT OBJECT ADDR: D3C4

ADDR  DESCRIPTION/CONTENTS

D3C4  Initialize checksum
D3C9  Read "odd" encoded byte 1X1X1X1X (C08C)
D3CE  Align "odd" bits 11X11X1
D3CF  Save for later (D36B)
D3D2  Read "even" encoded byte 1X1X1X1X (C08C)
D3D7  Combine bytes XXXXXXX (D36B)
D3DA  Preserve data (Volume,Track,sector,checksum) (D36D)
D3DB  Do checksum computation (D36C)
D3DE  Decrement counter - Finished field yet?
D3E1  No, do some more >>D3C6
D3E3  Is checksum computation zero?
D3E4  No, then exit with carry set >>D3FB
D3E6  Read data register (C08C)
D3E9  Loop until data valid >>D3E6
D3EB  Is it first trailing byte ($DE)?
D3ED  No, then exit with carry set >>D3FB
D3EF  Delay for data latch to clear
D3F0  Read data register (C08C)
D3F3  Loop until data valid >>D3F0
D3F5  Is it second trailing byte ($AA)?
D3F7  No, then exit with carry set >>D3FB
D3F9  Clear the Carry flag (no error)
D3FA  Return to caller
D3FB  Set the Carry flag (error occurred)
D3FC  Return to caller

D3FD ********** READ DATA (ON THE FLY) ROUTINE ***************

D3FD Convert slot number to an
D3FE absolute reference (i.e. $60 -> $EC)
D400 Modify code for current slot number (D45A)
D403 (i.e. $C08C,X -> $C0BC) (D473)
D40F Get data buffer pointers
D413 Modify code for current Buffer address (D4AP)
D416 Provides access to top 3rd of Buffer (D48B)
D417 Subtract $54 from current address
D41F Modify code for current address - $54 (D497)
D422 Provides access to middle 3rd of Buffer (D49B)
D426 Subtract $57 from current address
D42B Modify code for current address - $AB (D478)
D42E Provides access to bottom 3rd of Buffer (D471)

D431 Initialize must find count at $20
D432 Decrement count - More to do?
D434 No, then exit >>D46D
D436 Read data register (C08C)
D439 Loop until data valid >>D436
D43B Is it 1st header mark ($D5)?
D43D No, then try again >>D433
D43F Delay for register to clear
D440 Read data register (C08C)
<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>D443</td>
<td>Loop until data valid &gt;&gt;D440</td>
</tr>
<tr>
<td>D445</td>
<td>Is is 2nd header mark ($AA)?</td>
</tr>
<tr>
<td>D447</td>
<td>No, then see if it is last header mark &gt;&gt;D43B</td>
</tr>
<tr>
<td>D449</td>
<td>Delay for register to clear</td>
</tr>
<tr>
<td>D44A</td>
<td>Read data register (C08C)</td>
</tr>
<tr>
<td>D450</td>
<td>Loop until data valid &gt;&gt;D44A</td>
</tr>
<tr>
<td>D45F</td>
<td>Is is 3rd header mark ($AB)?</td>
</tr>
<tr>
<td>D451</td>
<td>No, then see if it is last header mark &gt;&gt;D43B</td>
</tr>
<tr>
<td>D453</td>
<td>Initialize offset into data buffer</td>
</tr>
<tr>
<td>D457</td>
<td>Initialize checksum</td>
</tr>
<tr>
<td>D459</td>
<td>Read a data byte (C08C)</td>
</tr>
<tr>
<td>D45E</td>
<td>Translate it (D100)</td>
</tr>
<tr>
<td>D461</td>
<td>Store it in Auxiliary buffer (D256)</td>
</tr>
<tr>
<td>D464</td>
<td>Compute running checksum</td>
</tr>
<tr>
<td>D466</td>
<td>Increment offset - More to do?</td>
</tr>
<tr>
<td>D467</td>
<td>Yes, then continue &gt;&gt;D457</td>
</tr>
<tr>
<td>D469</td>
<td>Reinitialize offset into data buffer</td>
</tr>
<tr>
<td>D472</td>
<td>Branch always taken &gt;&gt;D472</td>
</tr>
<tr>
<td>D47D</td>
<td>Set carry flag indicating error</td>
</tr>
<tr>
<td>D47E</td>
<td>Return to caller</td>
</tr>
<tr>
<td>D47F</td>
<td>Store byte in Primary buffer (bottom third) (1000)</td>
</tr>
<tr>
<td>D477</td>
<td>Translate it and merge in (D100)</td>
</tr>
<tr>
<td>D47A</td>
<td>bits from Auxiliary buffer (D256)</td>
</tr>
<tr>
<td>D480</td>
<td>Increment offset - done yet?</td>
</tr>
<tr>
<td>D481</td>
<td>No, then do another &gt;&gt;D46F</td>
</tr>
<tr>
<td>D483</td>
<td>Save last byte for later, no time now</td>
</tr>
<tr>
<td>D484</td>
<td>Strip off last two bits XXXXXX00</td>
</tr>
<tr>
<td>D486</td>
<td>Reinitialize offset</td>
</tr>
<tr>
<td>D488</td>
<td>Read a byte (C08C)</td>
</tr>
<tr>
<td>D48D</td>
<td>Translate it and merge in (D100)</td>
</tr>
<tr>
<td>D490</td>
<td>bits from Auxiliary buffer (D256)</td>
</tr>
<tr>
<td>D496</td>
<td>Store byte in Primary buffer (middle third) (1000)</td>
</tr>
<tr>
<td>D499</td>
<td>increment offset - done yet?</td>
</tr>
<tr>
<td>D49A</td>
<td>No, then do another &gt;&gt;D480</td>
</tr>
<tr>
<td>D49C</td>
<td>Read a byte (C08C)</td>
</tr>
<tr>
<td>D4A1</td>
<td>Strip off last two bits XXXXXX00</td>
</tr>
<tr>
<td>D4A3</td>
<td>Reinitialize offset</td>
</tr>
<tr>
<td>D4A5</td>
<td>Translate byte and merge in (D100)</td>
</tr>
<tr>
<td>D4A8</td>
<td>bits from Auxiliary buffer (D254)</td>
</tr>
<tr>
<td>D4AE</td>
<td>Store byte in Primary buffer (top third) (1000)</td>
</tr>
<tr>
<td>D4B1</td>
<td>Read a byte (C08C)</td>
</tr>
<tr>
<td>D4B6</td>
<td>Increment offset - done yet?</td>
</tr>
<tr>
<td>D4B7</td>
<td>No, then do another &gt;&gt;D4A5</td>
</tr>
<tr>
<td>D4B9</td>
<td>Strip off last two bits XXXXXX00</td>
</tr>
<tr>
<td>D4BB</td>
<td>Is checksum valid? (D100)</td>
</tr>
<tr>
<td>D4BE</td>
<td>No, then exit with error &gt;&gt;D4CC</td>
</tr>
<tr>
<td>D4C0</td>
<td>Get slot number</td>
</tr>
<tr>
<td>D4C2</td>
<td>Read data register (C08C)</td>
</tr>
<tr>
<td>D4C5</td>
<td>Loop until data valid &gt;&gt;D4C2</td>
</tr>
<tr>
<td>D4C7</td>
<td>Is is last trailing mark ($DE)?</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>D4CA</td>
<td>Yes, then continue with carry clear &gt;&gt;D4CD</td>
</tr>
<tr>
<td>D4CC</td>
<td>Set Carry flag indicating error</td>
</tr>
<tr>
<td>D4CD</td>
<td>Get byte we stored away, we have time now</td>
</tr>
<tr>
<td>D4CE</td>
<td>Set proper offset</td>
</tr>
<tr>
<td>D4D0</td>
<td>Store byte in Primary buffer (offset $55)</td>
</tr>
<tr>
<td>D4D2</td>
<td>Return to caller</td>
</tr>
<tr>
<td>D4D3</td>
<td>UPDATE DEVICE TRACK TABLE</td>
</tr>
<tr>
<td>D4D6</td>
<td>Update Device Track Table (D359)</td>
</tr>
<tr>
<td>D4D9</td>
<td>Return to caller</td>
</tr>
<tr>
<td>D4DA</td>
<td>DETERMINE IF DRIVE IS ON (DATA CHANGING)</td>
</tr>
<tr>
<td>D4DA</td>
<td>Get slot number</td>
</tr>
<tr>
<td>D4DC</td>
<td>Initialize counter</td>
</tr>
<tr>
<td>D4DE</td>
<td>Read data register (C08C)</td>
</tr>
<tr>
<td>D4E1</td>
<td>Delay 25 cycles (D4F0)</td>
</tr>
<tr>
<td>D4E6</td>
<td>Has data register changed? (C08C)</td>
</tr>
<tr>
<td>D4E9</td>
<td>Yes, then exit &gt;&gt;D4F0</td>
</tr>
<tr>
<td>D4EB</td>
<td>Just in case indicate No Device Connected Error</td>
</tr>
<tr>
<td>D4ED</td>
<td>Drive register (D100) - 256 tries yet?</td>
</tr>
<tr>
<td>D4EE</td>
<td>No, try again &gt;&gt;D4DE</td>
</tr>
<tr>
<td>D4F0</td>
<td>Return to caller</td>
</tr>
<tr>
<td>D4F1</td>
<td>CONVERT SLOT/DRIVE TO TABLE OFFSET</td>
</tr>
<tr>
<td>D4F1</td>
<td>Preserve A-register</td>
</tr>
<tr>
<td>D4F2</td>
<td>Get Unit number DSSS8000</td>
</tr>
<tr>
<td>D4F4</td>
<td>Divide by 16 000000000 D</td>
</tr>
<tr>
<td>D4F8</td>
<td>Put Drive into Carry 000000000 D</td>
</tr>
<tr>
<td>D4FA</td>
<td>Strip out Drive 000000000 D</td>
</tr>
<tr>
<td>D4FC</td>
<td>Roll left 000000000 D</td>
</tr>
<tr>
<td>D4FD</td>
<td>Put result in X-register</td>
</tr>
<tr>
<td>D4FE</td>
<td>Restore A-register</td>
</tr>
<tr>
<td>D4FF</td>
<td>Return to caller</td>
</tr>
<tr>
<td>D500</td>
<td>WRITE DATA ROUTINE</td>
</tr>
<tr>
<td>D500</td>
<td>Set Carry flag (anticipate error)</td>
</tr>
<tr>
<td>D504</td>
<td>Is diskette &quot;write-protected&quot;? (C08E)</td>
</tr>
<tr>
<td>D507</td>
<td>No, then continue on &gt;&gt;D50C</td>
</tr>
<tr>
<td>D509</td>
<td>Go to error routine &gt;&gt;D50F</td>
</tr>
<tr>
<td>D50C</td>
<td>Put transition byte from secondary buffer (D300)</td>
</tr>
<tr>
<td>D50F</td>
<td>into zero page for timing</td>
</tr>
<tr>
<td>D511</td>
<td>Use $FF for &quot;sync&quot; byte</td>
</tr>
<tr>
<td>D513</td>
<td>Write first &quot;sync&quot; byte (C08F)</td>
</tr>
<tr>
<td>D519</td>
<td>Set counter for four more</td>
</tr>
<tr>
<td>D51C</td>
<td>Delay so that writes occur</td>
</tr>
<tr>
<td>D51D</td>
<td>Exactly on 40 cycle loops</td>
</tr>
</tbody>
</table>
### Disk II Device Driver -- V1.1.1 -- 18 SEP 84

**NEXT OBJECT ADDR: D51E**

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>D51B</td>
<td>---</td>
</tr>
<tr>
<td>D520</td>
<td>Write &quot;sync&quot; byte (D5E7)</td>
</tr>
<tr>
<td>D523</td>
<td>Decrement counter, done yet?</td>
</tr>
<tr>
<td>D524</td>
<td>No, then do another &gt;&gt;D51E</td>
</tr>
<tr>
<td>D526</td>
<td>Write first data mark (D5)</td>
</tr>
<tr>
<td>D52B</td>
<td>Write second data mark ($AA)</td>
</tr>
<tr>
<td>D530</td>
<td>Write third data mark ($AD)</td>
</tr>
<tr>
<td>D535</td>
<td>Initialize checksum</td>
</tr>
<tr>
<td>D536</td>
<td>Initialize index into Auxiliary buffer</td>
</tr>
<tr>
<td>D538</td>
<td>Branch always taken &gt;&gt;D53D</td>
</tr>
<tr>
<td>D53A</td>
<td>Get data byte (Auxiliary buffer) (D300)</td>
</tr>
<tr>
<td>D53D</td>
<td>Exclusive-or with previous data byte (D2FP)</td>
</tr>
<tr>
<td>D540</td>
<td>Put result in X-reg for table lookup</td>
</tr>
<tr>
<td>D541</td>
<td>Lookup &quot;disk byte&quot; in table (D203)</td>
</tr>
<tr>
<td>D544</td>
<td>Get slot</td>
</tr>
<tr>
<td>D546</td>
<td>Write &quot;disk byte&quot; (C0BD)</td>
</tr>
<tr>
<td>D54C</td>
<td>Decrement index - Done with Auxiliary buffer?</td>
</tr>
<tr>
<td>D54D</td>
<td>No, then another byte &gt;&gt;D53A</td>
</tr>
<tr>
<td>D54F</td>
<td>Get last byte of Auxiliary buffer</td>
</tr>
<tr>
<td>D551</td>
<td>Initialize index into Primary buffer</td>
</tr>
<tr>
<td>D553</td>
<td>Exclusive-or with next data byte (1000)</td>
</tr>
<tr>
<td>D556</td>
<td>Strip out last two bits XXXXX00</td>
</tr>
<tr>
<td>D55B</td>
<td>Put result in X-reg for table lookup</td>
</tr>
<tr>
<td>D559</td>
<td>Lookup &quot;disk byte&quot; in table (D003)</td>
</tr>
<tr>
<td>D55C</td>
<td>Get slot</td>
</tr>
<tr>
<td>D55E</td>
<td>Write &quot;disk byte&quot; (C0BD)</td>
</tr>
<tr>
<td>D564</td>
<td>Get data byte (Primary buffer) (1000)</td>
</tr>
<tr>
<td>D567</td>
<td>Increment offset, end of this page?</td>
</tr>
<tr>
<td>D568</td>
<td>No, then continue on &gt;&gt;D553</td>
</tr>
<tr>
<td>D56A</td>
<td>Did buffer start on page boundary?</td>
</tr>
<tr>
<td>D56C</td>
<td>Yes, then go write checksum &gt;&gt;D5C0</td>
</tr>
<tr>
<td>D56D</td>
<td>Did buffer start one past page boundary?</td>
</tr>
<tr>
<td>D570</td>
<td>Yes, then go write last byte &gt;&gt;D53B</td>
</tr>
<tr>
<td>D572</td>
<td>Carry indicates odd or even buffer end</td>
</tr>
<tr>
<td>D573</td>
<td>Get transition byte</td>
</tr>
<tr>
<td>D575</td>
<td>Write it (C0BD)</td>
</tr>
<tr>
<td>D57B</td>
<td>Get second transition byte</td>
</tr>
<tr>
<td>D57D</td>
<td>Delay 2 cycles for correct timing</td>
</tr>
<tr>
<td>D57E</td>
<td>Increment offset, buffer end on odd byte?</td>
</tr>
<tr>
<td>D57F</td>
<td>Yes, go see if we're done then &gt;&gt;D599</td>
</tr>
<tr>
<td>D581</td>
<td>Exclusive-or with next data byte (1100)</td>
</tr>
<tr>
<td>D584</td>
<td>Strip out last two bits XXXXX00</td>
</tr>
<tr>
<td>D586</td>
<td>Put result in X-reg for table lookup</td>
</tr>
<tr>
<td>D587</td>
<td>Lookup &quot;disk byte&quot; in table (D203)</td>
</tr>
<tr>
<td>D58A</td>
<td>Get slot</td>
</tr>
<tr>
<td>D58C</td>
<td>Write &quot;disk byte&quot; (C0BD)</td>
</tr>
<tr>
<td>D592</td>
<td>Get data byte (Primary buffer - page 2) (1100)</td>
</tr>
<tr>
<td>D595</td>
<td>Increment offset</td>
</tr>
<tr>
<td>D596</td>
<td>Exclusive-or with next data byte (1100)</td>
</tr>
<tr>
<td>D599</td>
<td>End of buffer? - Put result in carry</td>
</tr>
<tr>
<td>D59B</td>
<td>Strip out last two bits XXXXX00</td>
</tr>
</tbody>
</table>

### Disk II Device Driver -- V1.1.1 -- 18 SEP 84

**NEXT OBJECT ADDR: D59D**

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>D59D</td>
<td>Put result in X-reg for table lookup</td>
</tr>
<tr>
<td>D59E</td>
<td>Lookup &quot;disk byte&quot; in table (D2B3)</td>
</tr>
<tr>
<td>D5A1</td>
<td>Get slot</td>
</tr>
<tr>
<td>D5A3</td>
<td>Write &quot;disk byte&quot; (C0BD)</td>
</tr>
<tr>
<td>D5A9</td>
<td>Get data byte (Primary buffer - page 2) (1100)</td>
</tr>
<tr>
<td>D5AC</td>
<td>Increment offset - Done yet?</td>
</tr>
<tr>
<td>D5AD</td>
<td>No, then do another &gt;&gt;D581</td>
</tr>
<tr>
<td>D5AF</td>
<td>Yes, checksum go write checksum &gt;&gt;D581</td>
</tr>
<tr>
<td>D5B1</td>
<td>--- &gt;&gt;D5C0</td>
</tr>
<tr>
<td>D5B3</td>
<td>Get last byte</td>
</tr>
<tr>
<td>D5B6</td>
<td>Write it (C0BD)</td>
</tr>
<tr>
<td>D5BC</td>
<td>Delay 14 cycles for correct timing</td>
</tr>
<tr>
<td>D5C0</td>
<td>Use last byte in Primary buffer as checksum</td>
</tr>
<tr>
<td>D5C2</td>
<td>Lookup &quot;disk byte&quot; (D2B3)</td>
</tr>
<tr>
<td>D5C5</td>
<td>Get slot</td>
</tr>
<tr>
<td>D5CD</td>
<td>Write &quot;disk byte&quot; (C0BD)</td>
</tr>
<tr>
<td>D5CF</td>
<td>Delay 11 cycles for correct timing</td>
</tr>
<tr>
<td>D5D3</td>
<td>Load &quot;epilog&quot; from table ($DB,$EA,$EB,$FF) (D1C4)</td>
</tr>
<tr>
<td>D5D6</td>
<td>Go write it (D5B9)</td>
</tr>
<tr>
<td>D5D9</td>
<td>Increment offset</td>
</tr>
<tr>
<td>D5DA</td>
<td>Done all four yet?</td>
</tr>
<tr>
<td>D5DC</td>
<td>No, then do another &gt;&gt;D5D3</td>
</tr>
<tr>
<td>D5DE</td>
<td>Clear Carry flag (no error)</td>
</tr>
<tr>
<td>D5DF</td>
<td>Select read mode (C0BD)</td>
</tr>
<tr>
<td>D5E5</td>
<td>Return to caller</td>
</tr>
</tbody>
</table>

#### D5E6

**WRITE A BYTE SUBROUTINE**

- D5E6: Wait 9 cycles before write
- D5E7: Wait 7 cycles before write
- D5E9: Put A-register in data register (C0BD)
- D5EC: And write data register (C0BC)
- D5EF: Return to caller

#### D5F0

**PRENIZE BLOCK ROUTINE**

- D5F0: Get buffer pointer
- D5F5: Add $2 to buffer address
- D5F7: To access top third of buffer >>D5FA
- D5FA: Store result in code below (D638)
- D601: Subtract $54 from buffer address
- D603: To access middle third of buffer >>D606
- D606: Store result in code below (D625)
- D60D: Subtract $AA from buffer address
- D60F: To access bottom third of buffer >>D612
- D612: Store result in code below (D61B)
- D618: Initialize offset
- D61A: Get data byte (bottom third) XXXXXXX (1000)
- D61D: Get last two bits 000000AB
- D61F: Put in X-reg for table lookup
<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>D620</td>
<td>Use lookup to reposition bits $000BAA0 (D1E9)</td>
</tr>
<tr>
<td>D623</td>
<td>Save result on stack</td>
</tr>
<tr>
<td>D624</td>
<td>Get data byte (middle third) $xxxxxxx (1056)</td>
</tr>
<tr>
<td>D627</td>
<td>Get last two bits $0000CD</td>
</tr>
<tr>
<td>D629</td>
<td>Put in X-reg for table lookup $000BAA0</td>
</tr>
<tr>
<td>D62A</td>
<td>Get current value from stack $000DCA0 (D1C0)</td>
</tr>
<tr>
<td>D62B</td>
<td>Merge in new bits using table $0000DCA0</td>
</tr>
<tr>
<td>D62E</td>
<td>Save result on stack</td>
</tr>
<tr>
<td>D62F</td>
<td>Get data byte (top third) $xxxxxxx (10AC)</td>
</tr>
<tr>
<td>D632</td>
<td>Get last two bits $0000EF</td>
</tr>
<tr>
<td>D634</td>
<td>Put in X-reg for table lookup</td>
</tr>
<tr>
<td>D635</td>
<td>Get current value from stack $000DCA0</td>
</tr>
<tr>
<td>D636</td>
<td>Merge in new bits using table $FEDCBA00 (D1A0)</td>
</tr>
<tr>
<td>D639</td>
<td>Save result on stack</td>
</tr>
<tr>
<td>D63A</td>
<td>Get offset into primary buffer</td>
</tr>
<tr>
<td>D63B</td>
<td>Compute offset into Auxiliary buffer</td>
</tr>
<tr>
<td>D63D</td>
<td>Put in X-reg</td>
</tr>
<tr>
<td>D63E</td>
<td>Get data byte just created $FEDCBA00</td>
</tr>
<tr>
<td>D63F</td>
<td>Store it in Auxiliary buffer (D300)</td>
</tr>
<tr>
<td>D642</td>
<td>Increment offset primary buffer, done yet?</td>
</tr>
<tr>
<td>D643</td>
<td>No, then do another $&gt;D61A</td>
</tr>
<tr>
<td>D645</td>
<td>Get low order byte of buffer</td>
</tr>
<tr>
<td>D647</td>
<td>Subtract 1 (offset to last byte in buffer)</td>
</tr>
<tr>
<td>D648</td>
<td>Save it for later</td>
</tr>
<tr>
<td>D64A</td>
<td>Get low order byte of buffer</td>
</tr>
<tr>
<td>D64C</td>
<td>Modify code in Write Data Routine (offset) (D552)</td>
</tr>
<tr>
<td>D64F</td>
<td>Buffer on page boundary? - Yes, skip ahead $&gt;D65F</td>
</tr>
<tr>
<td>D651</td>
<td>Else, compute offset to last byte</td>
</tr>
<tr>
<td>D653</td>
<td>Before page boundary</td>
</tr>
<tr>
<td>D654</td>
<td>Get byte (page boundary -1)</td>
</tr>
<tr>
<td>D656</td>
<td>Point at next byte (page boundary)</td>
</tr>
<tr>
<td>D657</td>
<td>Exclusive-or them together $xxxxxxx</td>
</tr>
<tr>
<td>D659</td>
<td>Strip off last two bits $xxxxx00</td>
</tr>
<tr>
<td>D65B</td>
<td>Put in X-reg for table lookup</td>
</tr>
<tr>
<td>D65C</td>
<td>Get &quot;disk byte&quot; from table (transition byte) (D203)</td>
</tr>
<tr>
<td>D65F</td>
<td>Save result ($ indicates page boundary)</td>
</tr>
<tr>
<td>D661</td>
<td>Buffer on page boundary? - Yes skip ahead $&gt;D66F</td>
</tr>
<tr>
<td>D663</td>
<td>Get offset to last byte in buffer</td>
</tr>
<tr>
<td>D665</td>
<td>Carry indicates odd or even buffer start</td>
</tr>
<tr>
<td>D666</td>
<td>Get byte (page boundary)</td>
</tr>
<tr>
<td>D668</td>
<td>Did buffer start on odd byte? - Yes skip $&gt;D66D</td>
</tr>
<tr>
<td>D66A</td>
<td>Point at next byte (page boundary +1)</td>
</tr>
<tr>
<td>D66B</td>
<td>Exclusive-or them together</td>
</tr>
<tr>
<td>D66D</td>
<td>Save result</td>
</tr>
<tr>
<td>D66F</td>
<td>Point at last byte in buffer</td>
</tr>
<tr>
<td>D671</td>
<td>Get last byte in buffer $xxxxxxx</td>
</tr>
<tr>
<td>D673</td>
<td>Strip off last two bits $xxxxx00</td>
</tr>
<tr>
<td>D675</td>
<td>Save result (&quot;checksum byte&quot;)</td>
</tr>
<tr>
<td>D677</td>
<td>Get high order byte of buffer</td>
</tr>
<tr>
<td>D679</td>
<td>Modify code in Write Data Routine (D555)</td>
</tr>
<tr>
<td>D680</td>
<td>Get slot number for this operation</td>
</tr>
<tr>
<td>D68E</td>
<td>Modify code in Write Data Routine (D55D)</td>
</tr>
<tr>
<td>D69A</td>
<td>Return to caller</td>
</tr>
<tr>
<td>D69B</td>
<td>***************** DETERMINE IF SLOT/DRIVE HAS CHANGED *****************</td>
</tr>
<tr>
<td>D69D</td>
<td>Compare unit number with &quot;current&quot; unit number (D359)</td>
</tr>
<tr>
<td>D69F</td>
<td>Put &quot;current&quot; drive in Carry</td>
</tr>
<tr>
<td>D69D</td>
<td>Has slot changed? - No, then exit $&gt;D6BD</td>
</tr>
<tr>
<td>D69D</td>
<td>Get &quot;current&quot; slot</td>
</tr>
<tr>
<td>D69A</td>
<td>Put in X-register</td>
</tr>
<tr>
<td>D69C</td>
<td>Exit if Slot $&gt;D6BD</td>
</tr>
<tr>
<td>D69D</td>
<td>Is &quot;current&quot; motor is on? (D4DC)</td>
</tr>
<tr>
<td>D69D</td>
<td>No, then exit $&gt;D6BD</td>
</tr>
<tr>
<td>D69D</td>
<td>Wait until &quot;current&quot; motor is off (D370)</td>
</tr>
<tr>
<td>D69B</td>
<td>Or else timeout $&gt;D6A6</td>
</tr>
<tr>
<td>D6BD</td>
<td>Return to caller</td>
</tr>
<tr>
<td>D68E</td>
<td>***************** CLEAR IWM PHASES *****************</td>
</tr>
<tr>
<td>D6BE</td>
<td>Get unit number</td>
</tr>
<tr>
<td>D6C8</td>
<td>Strip drive bit</td>
</tr>
<tr>
<td>D6C2</td>
<td>Put slot*16 in X-Register</td>
</tr>
<tr>
<td>D6C3</td>
<td>Clear phases in case there is (C080)</td>
</tr>
<tr>
<td>D6C6</td>
<td>one of them new-fangled storage (C082)</td>
</tr>
<tr>
<td>D6C9</td>
<td>devices sharing this slot (C084)</td>
</tr>
<tr>
<td>D6C0</td>
<td>with my (trusty old Disk II. (C086)</td>
</tr>
<tr>
<td>D6CF</td>
<td>Return to caller</td>
</tr>
<tr>
<td>D6D0</td>
<td>***************** CHECK CALLING PARAMETERS *****************</td>
</tr>
<tr>
<td>D6D0</td>
<td>Check command code</td>
</tr>
<tr>
<td>D6D2</td>
<td>Is it greater or equal to 47</td>
</tr>
<tr>
<td>D6D6</td>
<td>Get Block Number</td>
</tr>
<tr>
<td>D6DD</td>
<td>Is Block Number good? (D356)</td>
</tr>
<tr>
<td>D6DE</td>
<td>Yes, if less than $100 $&gt;D68</td>
</tr>
<tr>
<td>D6E0</td>
<td>No, if greater than or equal to $200 $&gt;D6E6</td>
</tr>
<tr>
<td>D6E4</td>
<td>No, if greater than or equal to $118 $&gt;D6EB</td>
</tr>
<tr>
<td>D6EB</td>
<td>Offset to last byte in buffer</td>
</tr>
<tr>
<td>D6EB</td>
<td>Indicate</td>
</tr>
<tr>
<td>D6EB</td>
<td>Return to caller</td>
</tr>
<tr>
<td>D6EA</td>
<td>All is well</td>
</tr>
<tr>
<td>D6EB</td>
<td>Return to caller</td>
</tr>
<tr>
<td>D6EA</td>
<td>Not used</td>
</tr>
</tbody>
</table>

$D6EA$-$D6FF$ NOT USED
**IRQ Handler -- V1.1.1 -- 18 SEP 84**

**ADDR** | **DESCRIPTION/CONTENTS**  
--- | ---  
**FF9B** | **MODULE STARTING ADDRESS**  
**--------------------------**  
* * IRQ Handler  
* * Resides at $FF9B. Put  
* * there by ProDOS Relocator.  
* *  
* VERSION 1.1.1 -- 18 SEP 84  
* * (The IRQ Handler is still the  
* * same as it was in Version 1.0.1)  
* *  
**--------------------------**

**FF9B ~~~~~~~~~~~~ GLOBAL PAGE EQUATES ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~**

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

**FF9B ~~~~~~~~~~~~ EXTERNAL EQUATES ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~**

*D800* | RAM/ROM test byte  
*C802* | ROM Select  
*C80B* | BANK1 Select

**FF9B ~~~~~~~~~~~~ IRQ CODE ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~**

*FF9B* | Put A-Register on stack  
*FF9C* | Get Accumulator value from $45  
*FF9E* | and save it ($B56)  
*FFA1* | Replace $45 with A-Register  
*FFA2* | since it may have been destroyed  
*FFA4* | Load Status register  
*FFA5* | Restore onto stack  
*FFA6* | Isolate B flag – Was it a BRK?  
*FFA8* | Yes, skip Interrupt stuff >>FFC2  
*FFA9* | Else, Check location $D800 ($D800)  
*FFAD* | Do we have RAM active  
*FFA1* | Yes, indicate so >>FFB3  
*FFB1* | Else, indicate ROM  
*FFB3* | Update Bank ID byte ($B8D)  
*FFB6* | Also save temporarily ($B57)  
*FFB9* | Push ($B5F0) address of  
*FFB8* | routine to bank in Ram and  
*FFBC* | call IRQ on the stack  
*FFBF* | Push a new P-Register on stack with  
*FFC1* | the Interrupt Disable flag set  
*FFC2* | Push ($FA41) address less 1 of  
*FFC4* | Monitor IRQ on the stack

**FFC8 ~~~~~~~~~~~~ Select ROM – execution continues in ROM (C082)**

*FFCB* | Push ($FA61) address less 1 of (FFD7)  
*FFCE* | Hardware Reset routine on to stack  
*FFD3* | Exit via select ROM code above >>FFC8  
*FFD6* | Address (~1) of Hardware Reset routine  

**FFD6 ~~~~~~~~~~~~ IRQ CODE ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~**

*FFD6* | Save Accumulator in Global page (BF3B)  
*FFDB* | Restore $45 with original value (BF56)  
*FFE0* | Select RAM (read & write) (C08B)  
*FFE3* | use BANK1 (C08B)  
*FFE5* | Get Bank ID byte (BF57)  
*FFE9* | Leave via Global Page IRQ exit code >>BFD3

**FFEC ~~~~~~~~~~~~ $FFEC-$FFF9 UNUSED ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~**

*FFEC* | These unused bytes are at $4FEC-$4FF9 when  
*FFF9* | loaded as part of the "PRODOS" file.

**FFFA ~~~~~~~~~~~~ VECTORS ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~**

*FFFA* | NMI Vector  
*FFFC* | Reset Vector  
*FFFE* | IRQ Vector
HOW "BASIC.SYSTEM" IS LOADED AND RELOCATED

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

   I-------------------I
   I                I
   I "BASIC.SYSTEM" I
   I 21 BLOCK FILE I
   I
   I(20 data blocks I
   I plus one index I--> I
   I block) I
   I
   I LS2800 I
   I
   I-------------------I

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

   I-------------------I
   I                I
   I $BF00 I
   I BI GLOBAL PAGE I
   I-------------------I
   I $BE00 I
   I INAMES OF OPEN FILES I
   I-------------------I
   I $BD00 I
   I
   I
   I
   I BASIC I
   I
   I INTERPRETER I
   I
   I (run location) I
   I
   I-------------------I
   I $9A00 I
   I

3. The BI Relocator searches for a "STARTUP" file in the same directory as "BASIC.SYSTEM". If found, it loads and executes the "STARTUP" program. Otherwise, it prints out a greeting and cold starts BASIC by jumping to the BASIC entry point at $BE00.
MODULE STARTING ADDRESS

2000

******** BASIC GLOBAL PAGE **********

BC7A  BASIC INTERPRETER VERSION NUMBER
BE00  BASIC INTERPRETER ENTRY POINT
BE03  81 COMMAND SCANNER (SYNTAX)
BE10  COUT VECTORS FOR EACH SLOT
BE28  KSWL VECTORS FOR EACH SLOT
BE3C  DEFAULT SLOT NO.
BE3D  DEFAULT DRIVE NO.
BEF8  HIMEM

******** SYSTEM GLOBAL PAGE ***********

BF00  MACHINE LANGUAGE INTERFACE ENTRY
BF30  LAST DEVICE USED
BF58  MEMORY MAP
BF99  SYSTEM TYPE FLAGS
BF9A  SLOTS WHICH CONTAINS CARDS WITH ROM
BF9D  IF 0, NO PREFIX ACTIVE
BFFD  INTERPRETER VERSION NUMBER

******** ROM ADDRESSES ***************

EB00  APPLESOFT ENTRY POINT
EF59  BRK HANDLER
FF2F  INIT SCREEN, MONITOR, ETC.
FC58  CLEAR SCREEN, HOME CURSOR
FEDD  STANDARD CHARACTER OUT
FE6F  CHARACTER OUTPUT TO SCREEN
FE84  SET NORMAL CHARACTER ATTRIBUTE

2000  BASIC INTERPRETER RELOCATOR ENTRY *********************

2000  JUMP OVER STARTUP FILENAME >>2047
200E  STARTUP FILENAME LENGTH (7)
2007  'STARTUP'
200E  ALLOW FOR 64 CHAR FILENAME
2047  $00 --> $2400
204B  $02 --> $9A00
2055  COPY 35 PAGES
2058  COPY INTERP TO HIGH MEMORY AT $9A00 <20C4>
205D  PAGE FOLLOWING INTERP IMAGE IS...
205F  BASIC GLOBAL PAGE IMAGE
2061  COPY THAT TO $BE00 <20C4>
2064  TO GET 40-COL DISPLAY, SEND A CTRL-U
2066  OUT THE NORMAL OUTPUT VECTOR. <FDED>
2069  SET NORMAL CHARACTER ATTRIBUTE <FE6F>
206C  INITIALIZE SCREEN/WINDOW <FF2F>
206F  CLEAR SCREEN/HOME CURSOR <FC58>
BI Relocator -- V1.1.1 -- 18 JUN 84

NEXT OBJECT ADDR: 2076

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>2076</td>
<td>SET BITMAP TO MARK LOWER 48K FREE (BF59)</td>
</tr>
<tr>
<td>207C</td>
<td>EXCEPT PAGES 0 AND 1 AND</td>
</tr>
<tr>
<td>207E</td>
<td>TEXT PAGES 4 THROUGH 7 (BF58)</td>
</tr>
<tr>
<td>2086</td>
<td>MARK $9000-$BFFFF IN USE.</td>
</tr>
<tr>
<td>2081</td>
<td>NO, THEN CAN'T RUN INTERP &gt;&gt;20B1</td>
</tr>
<tr>
<td>20A0</td>
<td>GOT AT LEAST 64K?</td>
</tr>
<tr>
<td>20A1</td>
<td>NO, THIS ONLY WORKS IN 64K &gt;&gt;20B1</td>
</tr>
<tr>
<td>20A6</td>
<td>SET MY CSDL/KSDL FOR INTERP INIT (221A)</td>
</tr>
<tr>
<td>20AE</td>
<td>COPY ALL 4 BYTES &gt;&gt;20A6</td>
</tr>
<tr>
<td>20A1</td>
<td>THEN GO TO BASIC COLDSTART &gt;&gt;E000</td>
</tr>
</tbody>
</table>

(WE WILL GET CONTROL AT $20D4 AGAIN)

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

20B2

20B3 PRINT "UNABLE TO EXECUTE BASIC SYSTEM" (223F)

20C3 ALLOW REBOOT IF RESET PRESSED (03F4)

20C2 GO TO SLEEP FOREVER >>20C2

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

20C4

20C5 COPY FROM $8/1

20C7 TO $2/3

20CA A PAGE AT A TIME >>20C4

20D0 COUNT PAGES

20D3 RETURN

20D4 ******* CSDL INTERCEPT / CONTINUE ****************

20D4 "J" APPLESOFT PROMPT?

20D6 NO...DON'T PRINT WHATEVER IT IS >>20D3

20D4 YES, APPLESOFT DONE SETTING UP (BE10)

20D5 POINT CSDL TO STANDARD OUTPUT

20E2 CHECK LAST DEVICE USED (BF30)

20E5 SET ONLINE PARAMETER TO THIS (2238)

20EB DRIVE ONE OR TWO? >>20BE

20EE STORE DEFAULT DRIVE (D) (BE3D)

20F2 ISOLATE SLOT FROM DEVICE NO.

20F7 AND STORE DEFAULT SLOT (S) (BE3C)

20FF GET SLOT BYTE SHOWING CARDS PRESENT (BF99)

2102 PICK OFF ITS BITS ONE BY ONE

2108 SET OUTVCS AND INVCES TO $C500 (BE19)

210B FOR ALL SLOTS WITH ROMS IN THEM (BE20)

2115 ---

211B SET HIMEM TO $9600

211D IN VARIOUS PLACES

2124 GOT A DEFAULT PREFIX? (BF9A)

BI Relocator -- V1.1.1 -- 18 JUN 84

NEXT OBJECT ADDR: 2127

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>2127</td>
<td>NO &gt;&gt;214E</td>
</tr>
</tbody>
</table>
| 2129 | YES, ML: GET PREFIX <BF00>
| 212F | ERROR? >>218B |
| 2136 | BACKSCAN PREFIX FOR ""S (0280) |
| 213B | AND COUNT THEM IN $21BE (223E) |
| 214E | ---
| 213F | FOR A COUNT OF SUBLEVELS >>2136 |
| 2146 | MORE THAN JUST VOLUME NAME? >>216P |
| 2148 | NO, ML: SET PREFIX <BF00>
| 214E | ML: ONLINE <BF00>
| 2154 | ERROR? >>218B |
| 2156 | GET VOLUME LENGTH (0281) |
| 215B | NONE THERE? >>218B |
| 215F | ADD ONE TO NAME LENGTH (0282) |
| 2164 | AND PREFIX IT WITH A "" (0281) |
| 2167 | ML: SET PREFIX <BF00>
| 216D | ERROR? >>218B |

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

216F ML: GET FILE_INFO <BF00>

2172 FIND "STARTUP" FILE

2175 ERROR? >>218B

217A SAVE LENGTH OF STARTUP FILE NAME (2236)

217D COPY NAME TO $200 (2065)

2186 FIRST COMMAND WILL BE "STARTUP"

218B CHECK NUMBER OF SUBLEVELS (223E)

2190 MORE THAN JUST VOLUME? >>2198

2192 ML: SET PREFIX <BF00>

2199 ANY STARTUP FILE NAME? (2236)

219B YES, SKIP MESSAGE >>21C1

219D SET TRUE KSDL <2209>

21A2 PRINT " PRODOS BASIC 1.1 " (2267)

21AD PRINT ", COPYRIGHT ... (2283)

21B6 SKIPT THREE LINES

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

21C1 ---

21C3 COPY WARMSTART JMP TO PAGE 3 (21FF)

21C9 AND COLDSTART (03D3)

21CC AND CTRL-Y (03F9)

21CF POINT & VECTOR (2286)

21D2 TO $E003 (CMD SCANNER) (03F5)

21DB COPY BKR HANDLER JMP ALSO (2282)

21B7 AND RESET JMP (03F2)

21F2 SET POWER-UP BYTE ACCORDINGLY (03F4)

21F7 SET APPLESOFT IN NON-TRACE MODE

21F9 GET INTERPRETER VERSION NUMBER, (B077)

21FC PUT IT IN SYSTEM GLOBAL PAGE. (B0FD)

21FF GO TO INTERPRETER >>B500
Beneath Apple ProDOS Supplement

BI Relocator -- V1.1.1 -- 18 Jun 84

NEXT OBJECT ADDR: 21FF

ADDRESS DESCRIPTION/CONTENTS

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

2202 BREAK HANDLER ADDRESS FOR PAGE 3
2204 RESET HANDLER IS BASIC INTERP
2206 APPLESOFT & GOES TO BI CMD SCANNER >>BE03

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

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

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

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

221E GET FILE INFO PARMLIST
221F FILE NAME IS AT $2006
2221 15 BYTES RESERVED FOR OTHER GET_FILE PARMS (NOT USED)
222B THIS BYTE NOT USED

2231 SET PREFIX PARMLIST
2232 FOR PREFIX AT $2234

2234 NULL PREFIX
2235 "/"

2236 SAVED LENGTH OF STARTUP FILE NAME

223F '*** UNABLE TO EXECUTE BASIC SYSTEM ***'
2267 'PRODOS BASIC 1.1'
2283 'COPYRIGHT APPLE, 1983-84'

22A3 ********** $22A3-$23FF NOT USED ****************************

22A3 NOT USED

BI Relocator -- V1.1.1 -- 18 Jun 84

NEXT OBJECT ADDR: 22A5

ADDRESS DESCRIPTION/CONTENTS

2400 ********** START OF BI IMAGE ****************************

2400 BASIC INTERP IMAGE
BASIC Interpreter (BI) -- V1.1.1 -- 18 JUN 84
NEXT OBJECT ADDR: 9A00

---

9A00 MODULE STARTING ADDRESS

******************************************************************************
* * PRODOS BASIC INTERPRETER (BI) * *
* THIS CODE STARTS IN THE THIRD * *
* BLOCK OF THE FILE BASIC.SYSTEM. * *
* IT PERFORMS COMMAND HANDLING * *
* FOR ALL BUILT-IN PRODOS COM- * *
* MANDS AND SUPPORTS BASIC'S FILE * *
* HANDLING. * *
* VERSION 1.1 -- 18 JUN 84 * *
* DISTRIBUTED WITH PRODOS V1.1.1 * *
******************************************************************************

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

0024 CURSOR HORIZONTAL
0028 SCREEN LINE BASE ADDR
0029
0033 MONITOR PROMPT CHARACTER
0036 CRT DISPLAY VECTOR (CSWL)
0038 KEYBOARD INPUT VECTOR (KSWL)
0039
003A SCRATCH POINTER AND LOOP COUNTER
003B SCRATCH POINTER AND LOOP COUNTER
003C
003D POINTER TO APPLESOF VARS
003F
0050 APPLESOF: LINE NUMBER
005B
0067 APPLESOF: START OF PROGRAM PTR
0068 APPLESOF: LOMEM (START OF VARS)
0069
006A APPLESOF: START OF ARRAY VARS PTR
006B
006C APPLESOF: START OF FREEAREA PTR
006D
006E APPLESOF: START OF STRINGS PTR
006F
0070 APPLESOF: HIMEM (END OF STRINGS)
0073
0075 APPLESOF: CURRENT LINE BEING EXECUTED
0076
0093 APPLESOF: ADDR OF LINE AFTER FINDLINE
009C

---

00AF APPLESOF: END OF PROGRAM PTR
00B0 APPLESOF: START OF PROGRAM PTR
00B9
00DE APPLESOF: PROGRAM LOCKED (PROTECTED)
00DB APPLESOF: ONERR ACTIVE FLAG
00DE APPLESOF: ONERR CODE
00F2 APPLESOF: TRACER ACTIVE FLAG
00F8 APPLESOF: INTERNAL STACK

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

0100 START OF 6502 STACK
0200 KEYBOARD INPUT LINE BUFFER
03F4 POWERON RESET FLAG

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

BE06 EXTERNAL COMMAND ENTRY TO BI
BE0C PRINT ERROR MESSAGE ENTRY TO BI
BE0F PRODOS ERROR CODE
BE10 OUTPUT VECTORS FOR ALL SLOTS
BE13 CURRENT OUTPUT VECTOR
BE12 CURRENT INPUT VECTOR
BE34 PRODOS INTERCEPT VECTORS (INPUT/OUTPUT)
BE38 BI'S INTERNAL REDIRECTION VECTORS
BE3C DEFAULT SLOT
BE3D DEFAULT DRIVE
BE3E A REGISTER SAVE AREA
BE3F X REGISTER SAVE AREA
BE40 Y REGISTER SAVE AREA
BE41 TRACE FLAG (APPLESOF TRACE ON/OFF)
BE42 IMMEDIATE COMMANDS=0, DEFERRED=1
BE43 EXEC FILE ACTIVE=0$0
BE44 READ FILE ACTIVE=0$0
BE45 WRITE FILE ACTIVE=0$0
BE46 READING PREFIX ACTIVE=0$0
BE47 DIRECTORY FILE BEING ACCESSED
BE49 FREE STRING SPACE DURING GARBAGE COLLECT
BE4A BUFFERED I/O BYTE COUNT
BE4B INDEX INTO INPUT COMMAND LINE
BE4C LAST OUTPUT CHAR TO PREVENT RECURSION
BE4D NUMBER OF OPEN NON-EXEC FILES
BE4E EXEC FILE BEING CLOSED FLAG
BE4F READ FILE IS TRANSLATED DIRECTORY
BE50 VECTOR TO EXTERNAL COMMAND HANDLER
BE52 LENGTH-1 OF EXTERNAL COMMAND STRING
BE53 COMMAND NUMBER
BE54 PARAMETERS ALLOWED FOR THIS COMMAND
(SEE BIT DEFINITIONS IN TABLE LATER)
BE56 PARAMETERS FOUND WITH THIS COMMAND
BASIC Interpreter (BI) -- V1.1.1 -- 18 JUN 84  NEXT OBJECT ADDR: 9A00
-----------------------------------------------------------------------
ADDR   DESCRIPTION/CONTENTS
-----------------------------------------------------------------------

(SAME BIT DEFINITIONS AS FOR PSR)

BE58   A KEYWORD VALUE
BE5A   B KEYWORD VALUE
BE5D   E KEYWORD VALUE
BE5F   L KEYWORD VALUE
BE61   S KEYWORD VALUE
BE62   D KEYWORD VALUE
BE63   P KEYWORD VALUE
BE65   R KEYWORD VALUE
BE68   @ KEYWORD VALUE
BE6A   T KEYWORD VALUE
BE6B   SLOT NUMBER FROM IN# OR PR#
BE70   ISSUE MLI CALL AND XLATE ERROR CODES
MLI PARM LIST FIELDS
BEA3   CREATE: ACCESS CODE
BEA4   CREATE: FILE ID
BEA5   CREATE: AUX ID
BEA7   CREATE: FILE KIND
BEB4   SET/GET FILE INFO: FARM COUNT
BEB7   SET/GET FILE INFO: ACCESS CODE
BEB8   SET/GET FILE INFO: FILE ID
BEB9   SET/GET FILE INFO: AUX ID
BEBB   SET/GET FILE INFO: FILE KIND
BEBC   SET/GET FILE INFO: BLOCKS USED
BEBE   SET/GET FILE INFO: MODIFY DATE/TIME
BEC7   ONLINE/GET/SET MARK/EOF/BUF: REF NUM
BEC8   ONLINE/GET/SET MARK/EOF/BUF: MARK/BUF
BECF   OPEN: SYSTEM BUFFER
BED0   OPEN: REF NUM RETURNED
BED2   NEWLINE: REF NUM
BED3   NEWLINE: NEW LINE CHAR (ALWAYS CR)
BED6   READ/WRITE: REF NUM
BED7   READ/WRITE: DATA ADDRESS
BED9   READ/WRITE: LENGTH OF DATA
BEDB   READ/WRITE: ACTUAL LENGTH TRANSMITTED
BEDC   CLOSE/FLUSH: REF NUM
BEB8   BASIC HIMEM VALUE

******** SYSTEM GLOBAL PAGE ************

BF03   QUIT VECTOR
BF30   LAST DEVICE USED
BF58   MEMORY UTILIZATION BIT MAP
BF94   OPEN FILE LEVEL
BF9A   PREFIX ACTIVE FLAG (IF NONZERO)

******** INPUT/OUTPUT LOCATIONS **********

BASIC Interpreter (BI) -- V1.1.1 -- 18 JUN 84  NEXT OBJECT ADDR: 9A00
-----------------------------------------------------------------------
ADDR   DESCRIPTION/CONTENTS
-----------------------------------------------------------------------

C000   KEYBOARD STROBE
C010   KEYBOARD STROBE CLEAR
CPPP   RESET I/O ROMS

********* APPLESOFT ROM LOCATIONS *********

D43F   APPLESOFT RESTART ENTRY
D61A   FIND LINE BY NUMBER IN APPLESOFT
D655   SET POINTERS IN APPLESOFT
D702   EXECUTE NEW APPLESOFT STATEMENT
D820   APPLESOFT CMD EXECUTE
D865   APPLESOFT SIGNAL ERROR
ED24   APPLESOFT PRINT DECIMAL NUMBER
F273   APPLESOFT SET NORMAL CHARS

********** MONITOR ROM LOCATIONS **********

PC58   MONITOR CLEAR SCREEN/HOME CURSOR
F9C9   MONITOR CLEAR TO EOL
FD10   MONITOR READ KEY (NO CURSOR)
FDDE   COUT VECTOR

9A00   ************ BASIC INTERPRETER LOAD POINT **********************
(ENTRY POINT IS AT $ABF1, WARMDO)

9A00   *************** REMOVE KSWL/CSWL INTERCEPTS ********************

9A00   --
9A01   REPLACE CSWL/KSWL WITH CURRENT (BE38)
9A04   ACTUAL DEVICE DRIVER VECTORS
9A16   RETURN

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

9A17   SET IMMEDIATE COMMAND MODE
9A19   AND GO SET I/O VECTORS <9F76>
9A1C   KSWL/H ALREADY SET?
9A21   NO? THEN CHECK CSWL >>9A26
9A23   YES, CONTINUE >>9A33
9A26   CSWL/H ALREADY SET?
9A2B   YES, CONTINUE >>9A33
9A2D   NO, SAVE CURRENT INTERCEPTS FIRST >>9A8D

9A2F   ************** OUTPUT INTERCEPT: MODE = 0 ***********************
(IMMEDIATE MODE)
<table>
<thead>
<tr>
<th>BASIC Interpreter (Bl) -- V1.1.1 -- 18 JUN 84</th>
<th>NEXT OBJECT ADDR: 9A2F</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADDR</td>
<td>DESCRIPTION/CONTENTS</td>
</tr>
<tr>
<td>9A2F</td>
<td>&quot;#&quot; CHARACTER? (9F61)</td>
</tr>
<tr>
<td>9A32</td>
<td>NO... &gt;&gt;9A54</td>
</tr>
<tr>
<td>9A34</td>
<td>ELSE, SAVE X REG (BE3F)</td>
</tr>
<tr>
<td>9A38</td>
<td>CHECK STACK FOR $D812 AS RETURN ADDR (0103)</td>
</tr>
<tr>
<td>9A3B</td>
<td>(APPLESOFT TRACE, PRINTING #LINENO)</td>
</tr>
<tr>
<td>9A44</td>
<td>NOT TRACING? &gt;&gt;9A6E</td>
</tr>
<tr>
<td>9A46</td>
<td>ELSE, SET DEFERRED MODE=4</td>
</tr>
<tr>
<td>9A4B</td>
<td>GET SET TO PRINT THE &quot;#&quot; (9F61)</td>
</tr>
<tr>
<td>9A4E</td>
<td>RESTORE X REG (BE3F)</td>
</tr>
<tr>
<td>9A51</td>
<td>AND GO TO OTHER OUTPUT HANDLER &gt;&gt;B7F1</td>
</tr>
<tr>
<td>9A54</td>
<td>NOT A #, SAME AS LAST OUTPUT THG? (BE4C)</td>
</tr>
<tr>
<td>9A57</td>
<td>(SAVE FOR NEXT TIME THRU) (BE4C)</td>
</tr>
<tr>
<td>9A5A</td>
<td>NO, ALL IS WELL &gt;&gt;9A74</td>
</tr>
<tr>
<td>9A5C</td>
<td>TWO RETURNS IN A ROW?</td>
</tr>
<tr>
<td>9A5E</td>
<td>NO, ALL IS WELL &gt;&gt;9A74</td>
</tr>
<tr>
<td>9A60</td>
<td>HAS HORIZONTAL CURSOR POSN CHANGED?</td>
</tr>
<tr>
<td>9A62</td>
<td>YES... &gt;&gt;9A69</td>
</tr>
<tr>
<td>9A64</td>
<td>ELSE, ANYTHING IN PATHNAME BUFFER? (BCBD)</td>
</tr>
<tr>
<td>9A67</td>
<td>(MUST BE ALPHA)</td>
</tr>
<tr>
<td>9A69</td>
<td>RESTORE A REG</td>
</tr>
<tr>
<td>9A6B</td>
<td>PATHNAME IS THERE... &gt;&gt;9A74</td>
</tr>
<tr>
<td>9A6D</td>
<td>ELSE, WE ARE RECURRING INFINITELY, EXIT!</td>
</tr>
<tr>
<td>9A6E</td>
<td>WE WERE`NT TRACING AFTER ALL, RESTORE X (BE3F)</td>
</tr>
<tr>
<td>9A71</td>
<td>AND A REGS, THEN FALL THRU TO EXIT (9F61)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>9A74</td>
<td>ECHO OUTPUT CHAR AND EXIT ***************</td>
</tr>
<tr>
<td>9A74</td>
<td>PUT BACK REAL CSWL/KSWL VECTORS &lt;9A00&gt;</td>
</tr>
<tr>
<td>9A77</td>
<td>OUTPUT THE CHARACTER &lt;FDED&gt;</td>
</tr>
<tr>
<td>9A7A</td>
<td>WAS IT A RETURN?</td>
</tr>
<tr>
<td>9A7C</td>
<td>NO, EXIT NOW &gt;&gt;9A8D</td>
</tr>
<tr>
<td>9A7E</td>
<td>ELSE, WAS APPLESOFT TRACING?</td>
</tr>
<tr>
<td>9A82</td>
<td>YES &gt;&gt;9A8B</td>
</tr>
<tr>
<td>9A84</td>
<td>NO, CLEAR MY TRACE FLAG (PSEUDO TRACE NOW) (BE41)</td>
</tr>
<tr>
<td>9A87</td>
<td>FORCE APPLESOFT TO TRACE FOR MY BENEFIT ONLY</td>
</tr>
<tr>
<td>9A8B</td>
<td>RESTORE A REG AND FALL THRU TO EXIT BI</td>
</tr>
</tbody>
</table>

| 9A8D | SAVE ACTUAL IN/OUT VECTORS ****************** |

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>9A8E</td>
<td>COPY KSWL/H TO VECIN</td>
</tr>
<tr>
<td>9A98</td>
<td>AND CSWL/H TO VECOUT</td>
</tr>
<tr>
<td>9A9A</td>
<td>IN BI GLOBAL PAGE (BE31)</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>BASIC Interpreter (Bl) -- V1.1.1 -- 18 JUN 84</th>
<th>NEXT OBJECT ADDR: 9AA3</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADDR</td>
<td>DESCRIPTION/CONTENTS</td>
</tr>
<tr>
<td>9AA3</td>
<td>************ SET CSWL/KSWL INTERCEPTS ***************</td>
</tr>
<tr>
<td>9AA4</td>
<td>---</td>
</tr>
<tr>
<td>9AA7</td>
<td>COPY VDO/ISIO VECTORS (BE34)</td>
</tr>
<tr>
<td>9A97</td>
<td>TO CSWL</td>
</tr>
<tr>
<td>9AB1</td>
<td>AND KSWL</td>
</tr>
<tr>
<td>9AB9</td>
<td>EXIT TO CALLER</td>
</tr>
</tbody>
</table>

| 9ABA | INPUT INTERCEPT: MODE = 0 *********************** |

| 9ABA | IMMEDIATE MODE |

| 9AB0 | IS EXEC FILE ACTIVE? (BE43) |
| 9ABD | NO >>9AC5 |
| 9ABF | YES, SAVE REGISTERS <9F62> |
| 9AC2 | AND GO READ EXEC FILE FOR INPUT COMMANDS >>9BAF |
| 9AC5 | NO EXEC FILE, RESTORE REAL CSWL/KSWL <9A00> |
| 9AC8 | NO, READ A KEY FROM KEYBOARD <FD10> |
| 9ACB | RETURN! |
| 9ACD | NO, EXIT >>9AEB |
| 9ACF | YES, SAVE REGISTERS <9F62> |
| 9AD2 | STORE IT IN LINE BUFFER (B200) |

| 9AD5 | --- THIS ENTRY CALLED BY EXEC TO PROCESS A COMMAND STRING STORED AT $200 |
| 9ADB | GO PROCESSES COMMAND STRING <A677> |
| 9ADD | CHECK COMMAND NUMBER RETURNED FROM PARSSE (BE53) |
| 9AEB | EXIT BI RIGHT NOW? >>9AEB |
| 9AFA | NO, COMMAND RETURNED WITH ERROR CODE? >>9AF0 |
| 9AF0 | NO, RESTORE Y REG (BE4B) |
| 9AF1 | RETURN A BACKSPACE TO CALLER OF KEYBOARD |
| 9AF4 | AND A LINE INDEX OF ZERO |
| 9AE6 | EXIT THE BI >>9AEB |
| 9AEB | RESTORE CALLER`S REGISTERS <9F6C> |
| 9AEB | AND EXIT BI BY INSTALLING INTERCEPTS >>9A8D |

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

| 9AEE | "NO DEVICE CONNECTED" |
| 9AF0 | MAIN ENTRY; STORE ERROR CODE (BE0F) |
| 9AF3 | AND IN APPLESOFT ONERR |
| 9AF5 | CHECK BI STATE (BE42) |
| 9AFB | MEMORIZE WHETHER IT`S IMMEDIATE MODE |
| 9AFD | SET A HIGH FILE LEVEL FOR NON-EXEC FILES (BF94) |
| 9B02 | NO ACTIVE READ/WRITE FILES OR PREFIX READ (BE44) |
| 9B08 | CLOSE ALL OPEN FILES AT OR ABOVE (BEDE) |
| 9B0E | FILE LEVEL = 50F |
| 9B10 | MLI: CLOSE (ALL) <BE70> |
| 9B13 | ERROR? >>9B27 |
| 9B15 | WRITE ANY DATA I HAVE BUFFERED <A00> |
| 9B18 | ERROR? >>9B27 |
BASIC Interpreter (BI) -- V1.1.1 -- 18 JUN 84  
NEXT OBJECT ADDR: 9B1A

ADDR  DESCRIPTION/CONTENTS

9B1A  PUT FILE LEVEL BACK TO ZERO
9B22  NOW FLUSH ALL OPEN FILES
9B24  ML1: FLUSH (ALL) <BE70>
9B27  ---
9B28  ASSUME MODE WILL BE 4 (DEFERRED)
9B2A  MEMORIZE WHETHER BASIC ONERR ACTIVE
9B2C  DEFERRED MODE CURRENTLY? >>9B30
9B2E  NO, STILL IMMEDIATE MODE (MODE=0)
9B30  ---
9B31  SET MODE AS DEFINED ABOVE <9F76>
9B34  RESTORE BI'S CSW/L/KSWL INTERCEPTS <9AA3>
9B37  GET ERROR CODE (BE0F)
9B3B  BASIC ONERR ACTIVE? THEN GO HANDLE IT >>9B4D
9B3E  NO, JUST PRINT ERROR MESSAGE <BEC0>
9B41  CLOSE EXEC FILE IF ONE IS OPEN <B2FB>
9B45  DEFERRED MODE? >>9B53
9B47  IMMED. MODE, PRINT RETURN AND... <9FAB>
9B4A  WARMSTART APPLESOFT >>D43F
9B4D  RESTORE STACK FOR BASIC
9B52  PASS ERROR CODE TO BASIC
9B53  ---
9B55  JUMP INTO APPLESOFT ERROR HANDLER >>D865

9B58  ********** RETURN TO IMMED. MODE ***********************

9B58  CLEAR APPLESOFT ERRNUM
9B5C  WILL LOOK FOR "" FROM APPLESOFT
9B61  SET NORMAL VIDEO IN APPLESOFT <F273>
9B64  RESTORE TRUE CSW/L/KSWL <9A00>
9B67  TRY TO WRITE BUFFERED DATA <9FF4>
9B6A  RESET MODE/SET UP BI'S INTERCEPTS <9A17>
9B6D  RESTORE REGISTERS <9F6C>
9B70  GO TO PROCESS IMMED. INPUT REQUEST >>9ABA

9B73  ********puted INTERCEPT: MODE=4 OR 8 ************

9B73  SAVE REGISTERS <9F62>
9B76  PREFIX INPUT ACTIVE? (BE46)
9B79  NO >>9B7E
9B7B  YES, GO DO SPECIAL HANDLING >>9D67
9B7E  ELSE, IS READ FILE ACTIVE? (BE44)
9B81  NO >>9B86
9B83  YES, GO DO SPECIAL HANDLING FOR THAT >>9C16
9B86  ELSE, IS EXEC FILE ACTIVE? (BE43)
9B89  NO >>9B8F
9B8B  YES, GET PROMPT CHARACTER
9B8D  IT BETTER NOT BE A "$"
9B8F  IT IS, RETURN TO IMMEDIATE MODE >>9B58
9B91  ELSE, SET TRUE CSW/L/KSWL <9A00>
9B94  AND PASS CALLER'S AREG TO REMOVE CURSOR (BE3E)

9B97  RESTORE Y-REGISTER (BE40)
9B99  SAVE CURSOR AND GET A KEYPRESS <FD1D>
9B9D  BACKSPACE?
9B9F  NO, EXIT BI >>9BAC
9BA1  YES, CHECK PROMPT
9BA3  IF ITS A "",...
9BA5  THEN EXIT WITH THE BACKSPACE >>9BAA
9BAB  ELSE, IF AT START OF LINE, REPROMPT >>9B94
9BAA  MIDDLE OF LINE, RETURN A BACKSPACE
9BAC  EXIT BI TO CALLER >>9B8D

9BAF  ************** READ EXEC FILE **************

9BAF  REMOVE CURSOR FROM SCREEN
9B11  CHECK PROMPT CHARACTER
9B13  IF ITS A "",...
9B15  DO THINGS DIFFERENTLY >>9BF2
9B17  CHECK KEYBOARD (C000)
9B18  NO KEY READY? >>9BCD
9B1C  GOT A KEY, IS IT CONTROL-C?
9B1E  NO, IGNORE IT >>9BCD
9B20  YES, CLOSE EXEC FILE <B2FB>
9B23  IMMEDIATE MODE? (BE42)
9B26  NO >>9CB1
9B28  YES, CLEAR KEYBOARD STORE (C010)
9B2B  AND GO START NEW LINE >>9CB1
9B2D  SET UP FOR EXEC LINE READ <9D8A>
9B2D  READ A LINE TO $200 <9C6C>
9B2D  ERROR? >>9BFA
9B2F  SAVE REGISTERS <9F62>
9B30  HOP INTO LOOP >>9BDE
9B3A  ---
9B3B  BACKSCANNING $200 BUFFER (0200)
9B3E  FORCING THE MSB ON
9B3F  RESTORE TRUE CSW/L/KSWL <9A00>
9B40  GO PROCESS COMMAND LINE <9AD5>
9B4C  CHECK COMMAND NUMBER (BE53)
9B4F  IMMEDIATE EXIT? IF NOT, GET NEXT LINE >>9BCD
9B51  RETURN

********** HANDLE EXEC PROMPT > **********

9B52  GET SET TO READ EXEC LINE <9D8A>
9B55  READ SINGLE CHARACTER PER CALL <9C40>
9B58  NO ERRORS, EXIT TO CALLER NOW >>9B81

********** EXEC ERROR RECOVERY **********
BASIC Interpreter (B1) -- V1.1.1 -- 18 JUN 84 NEXT OBJECT ADDR: 9BFA

ADDR DESCRIPTION/CONTENTS

9BFA CLOSE EXEC FILE <B245>
9BFD WAS ERROR "END OF DATA"?
9BFF NO, REAL ERROR THEN >>9C13
9C01 ELSE, OK -- JUST STOP EXECING
9C03 GET CURSOR HORIZONTAL POSITION
9C05 IF IN MID LINE, PASS SCREEN CHAR BACK >>9C0E
9C07 ELSE, CHANGE PROMPT TO ""
9C0B AND RETURN WITH A BACKSPACE
9C0D RETURN
9C0E GET SCREEN CHARACTER UNDER CURSOR
9C10 AND EXIT THRU KSWL TO GET REAL KEYPRESS >>9B38
9C13 REAL ERROR, GO TO B1'S MAIN ERROR HANDLER >>9AF0

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

9C16 GET PROMPT
9C18 IF ITS A ""
9C1C THEN RESET TO IMMEDIATE MODE >>9B58
9C1F ELSE, REMOVE CURSOR FROM SCREEN (BE3E)
9C24 CHECK KEYBOARD (CB88)
9C27 NO KEYPRESS? >>9C31
9C29 GOT A KEY, IS IT CONTROL-C?
9C2B NO, IGNORE IT >>9C31
9C2D CLEAR STROBE AND EXIT TO CALLER (C910)
9C30 RETURN
9C31 GET PROMPT AGAIN
9C33 IS THIS A DIRECTORY FILE? (BE47)
9C36 YES >>9C95
9C38 NO, IS PROMPT = ""?
9C3A YES, READ A SINGLE BYTE AT A TIME >>9C42
9C3C ELSE, READ ENTIRE LINE <9C67>
9C3F ERROR? >>9C13
9C41 RETURN
9C42 READ SINGLE BYTE FROM INPUT FILE <9C48>
9C45 ERROR? >>9C13
9C47 RETURN

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

9C48 SAVE CURRENT READ/WRITE COUNT (BED9)
9C4B IN L KEYWORD VALUE (BE5F)
9C50 SET UP TO READ ONE BYTE (BED9)
9C53 MLI; READ <BE70>
9C58 ERROR? >>9C66
9C5A PUT COUNT BACK TO MAXIMUM AGAIN (BE5F)
9C60 GET FIRST CHARACTER ON $280 LINE (BED7)
9C63 AND RETURN THAT TO CALLER (B260)
9C66 RETURN

BASIC Interpreter (B1) -- V1.1.1 -- 18 JUN 84 NEXT OBJECT ADDR: 9C66

ADDR DESCRIPTION/CONTENTS

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

9C67 REMOVE CURSOR FROM SCREEN (BE3E)
9C6C ----
9C6E MLI; READ <BE70>
9C71 ERROR? >>9C66
9C73 GET LENGTH ACTUALLY TRANSMITTED (BEDB)
9C76 NOTHING? >>9C8E
9C79 GOT SOMETHING, FIND END OF DATA (BE7D)
9C7D FETCH LAST BYTE OF LINE (B1FF)
9C82 IS IT A RETURN CHARACTER?
9C84 NO, LEAVE LINE ALONE >>9C8E
9C86 YES, WAS L KEYWORD GIVEN? (BE57)
9C88 YES, LEAVE IT BE >>9C8E
9C8D ELSE, CHOP OFF THE RETURN ITSELF
9C8E AND EXIT WITH A RETURN
9C90 RESTORING Y REG AS YOU GO (BE40)
9C94 RETURN

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

9C95 "" PROMPT?
9C97 YES, EXIT RIGHT NOW >>9CB8
9C99 ELSE, REMOVE CURSOR FROM SCREEN (BE3E)
9CB E SET 80 COLUMNS
9CAB MLI; GET MARK <BE70>
9CAB ERROR? >>9D1F
9CAA ARE WE AT BEGINNING OF THIS FILE? (BB9C)
9CB0 NO, CONTINUE >>9CDF
9CB2 YES, CNT FLAG = 2
9CB7 READ DIRECTORY HEADER <B15D>
9CB8 ERROR? >>9D1F
9CBC REF NUM TIMES 32 (BED6)
9CC7 SET THE L VALUE OF THIS DIR FILE IN (BCFF)
9CC9 THE OPEN FILE LIST TO THE ENTRY LENGTH (BEB8)
9CCD AND THE NUMBER OF ENTRIES PER BLOCK (BD60)

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

9CD0 GO FORMAT NAME OF DIRECTORY <B888>
9CD3 STORE THE LENGTH OF LINE AT $280
9CD6 PUT A RETURN CHAR AT END OF LINE
9CDD AND EXIT TO CALLER
9CDE RETURN
9CDF GET CAT FLAG (BE4F)
9CE2 IF ZERO, GO PROCESS INDIVIDUAL ENTRIES >>9D22
9CE4 IF MINUS, GO DO SUMMARY LINE OR EXIT >>9CF9
9C66 POSITIVE, ASSUME NULL LINE WANTED
9C6B DROP CAT FLAG BY ONE (BE4F)
BASIC Interpreter (BI) -- V1.1.1 -- 18 JUN 84  NEXT OBJECT ADDR: 9CEB

9CEB IF ZERO, JUST GO PRINT A BLANK LINE >>9CD3

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

9CED ELSE, BLANK OUT $200 AND <A66C>
9CF2 UNPACK "NAME TYPE BLOCKS ETC..." <9F80>

9CF5 LINE LENGTH IS 80
9CF7 GO RETURN IT TO CALLER >>9CD3

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

9CF9 DO SUMMARY LINE?
9CB8 NO, JUST EXIT (ALL DONE) >>9D1C
9CBD YES, DROP CAT FLAG SO EXIT NEXT TIME (BE4F)
9DB2 CLEAR READ/WRITE COUNT (BED9)
9DBA MLI: READ <BE70>
9DBD FORMAT BLOCKS FREE AND INUSE SUMMARY LINE <B0E7>
9D11 GET REF NUM (BED6)
9D14 AND COPY TO GET/SET LIST (BEC7)
9D18 NO ERRORS, EXIT >>9CF5
9D1A ERROR, JUMP TO BI ERROR EXIT >>9D1F
9D1C "END OF DATA" ERROR
9D1F GO TO BI ERROR EXIT >>9AF0

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

9D22 SET DIR ENTRY NUM COUNTER TO -1
9D27 GET REF NUM (BED6)
9D2A -32
9D2F USE AS INDEX TO GET ENTRY LENGTH (BCFP)
9D35 AND ENTRIES PER BLOCK FROM OPEN FILE LIST (BD00)
9D3B POSITION ON EVEN BLOCK BOUNDARY (BEC9)
9D41 AND GET SECTOR OFFSET (BECB)
9D45 SKIP FILE/DIR ENTRIES UNTIL POSITIONED TO (BCBB)
9D48 CURRENT POSITION IN THIS BLOCK (BC37)
9D50 READ NEXT DIR ENTRY FROM FILE <B1D1>
9D53 NO ERROR? >>9D61
9D55 ERROR, IF RANGE ERROR...
9D57 NO, TRUE ERROR >>9D1F
9D59 RANGE ERROR, READY FOR SUMMARY LINE NEXT (BE4F)
9D5E RETURN A BLANK LINE THIS TIME >>9CD3

9D61 FORMAT FILE/DIR ENTRY INTO $261 <A4C4>
9D64 AND RETURN IT TO CALLER >>9CF5

BASIC Interpreter (BI) -- V1.1.1 -- 18 JUN 84  NEXT OBJECT ADDR: 9D67

9D67 ********** PREFIX INPUT ACTIVE ***********************

9D67 PROMPT = "J"?
9D6B NO, ALL IS WELL >>9D6E
9D6E YES, RETURN TO IMMEDIATE MODE NOW >>9B58
9D75 REMOVE CURSOR FROM SCREEN (BE3E)
9D75 PREFIX NO LONGER ACTIVE AFTER THIS (BE46)
9D78 COPY PATHNAME BUFFER (PREFIX) (BCBC)
9D7E TO $200 (01FF)
9D84 RETURN WITH IT TO BASIC (BCBC)
9D89 RETURN

9D8A ********** SETUP TO READ LINE FROM EXEC *********************

9D8A SET READ REF NUM FOR EXECL FILE (BCA3)
9D90 READ TO $26F
9D95 FOR SEC BYTES OF LENGTH
9D9A (OR UNTIL A RETURN CHAR)
9DAA RETURN

9DA3 ********** OUTPUT INTERCEPT; MODE = C ***********************

(LOOK FOR CONTROL-D)

9DA3 SAVE REGISTERS <9F62>
9DA6 PRINTING A CONTROL-D?
9DAB NO >>9DC1
9DAA YES, WRITE OUT ANY BUFFERED DATA <<F94>
9DAD NOTHING IN COMMAND LINE (BE48)
9DB0 READ FILE INACTIVE (BE44)
9DB3 WRITE FILE INACTIVE (BE45)
9DB6 PREFIX READ INACTIVE (BE46)
9DBB SET MODE = 8 FROM NOW ON <9F76>
9DBE RESTORE REGS AND EXIT >>9F6C
9DCA GOT A CONTROL-D...
9DC3 SET MODE = 4 FROM NOW ON <9F76>
9DC5 RESTORE REGISTERS <9F6C>
9DCA OUTPUT CHARACTER AND EXIT >>B7F1

9DC3 ********** OUTPUT INTERCEPT; MODE = B ****************************

(ASSEMBLE COMMAND LINE)

9DC3 SAVE REGISTERS <9F62>
9DD2 SAVE CHARS IN COMMAND LINE (0200)
9DD5 WAS IT A RETURN?
9DD7 YES, READY TO ROLL >>9DE7
9DD9 NO, BUMP CHARACTER COUNTER (BE4B)
9DE3 AND EXIT TO CALLER >>9DE3
9DE5 OOPS? LINE TOO LONG
9DE0 "SYNTAX ERROR" >>9AF0
9D8E ELSE, RESTORE X REG AND EXIT (BE3F)
BASIC Interpreter (BI) -- V1.1.1 -- 18 JUN 84

ADDR DESCRIPTION/CONTENTS

9DE6 9DE7 ----
9DE9 NULL LINE? >>9DF6
9DEB NO, PUT BACK TRUE CSWL/KSWL <9A09>
9DEE SYNTAX SCAN CMD LINE <A677>
9DF1 ERROR? >>9DE0
9DF3 NO, PUT BACK BI'S INTERCEPTS <9A8D>
9DF6 ---
9DF8 MODE = 4 NOW <9F76>
9DFB RESTORE REGS AND EXIT >>9F6C

9DFE *********** WRITE BUFFERED CHARACTER ***********************
9DFE SAVE Y REG (BE40)
9EE1 CHECK PROMPT
9EE3 CHECK TO SEE IF WE ARE IN "IF", >>9E11
9EE6 "PRINT", "LIST", OR "CALL" STATEMENTS >>9E11
9EE9 OF AN APPLESOFT PROGRAM >>9E11
9EEB IF NOT, EXIT TO CALLER... (BE48)
9EEE WITH CHARACTER ECHOED TO SCREEN >>9A74
9E11 GET INDEX TO TEMPORARILY BUFFERED CHARS (BE4A)
9E16 STORE INTO BUFFER JUST ABOVE HIMEM
9E1B BUMP INDEX (BE4A)
9E1E OK >>9E28
9E20 BUFFER FULL, SAVE REGISTERS <9F62>
9E23 WRITE BUFFER OUT TO DISK <9EE>
9E26 ERROR? >>9E08
9E28 RESTORE REGISTERS <9F6C>
9E2B AND EXIT ANYWAY

9E2C ********** OUTPUT INTERCEPT: MODE = 4 *********************
(INITIAL ENTRY FOR A RUNNING PROGRAM)
(FLUSH OUT NON COMMAND LINES)
9E2C PRINTING A "#"? (9F61)
9E2F NO >>9E49
9E31 YES, SAVE X REGISTER (BE3F)
9E35 RETURN ADDR IS IN APPLESOFT... (0103)
9E38 TRACE ROUTINE...
9E3C AT $8812? (0104)
9E41 YES >>9EB6
9E43 NO, RESTORE REGISTERS (9F61)
9E49 IS WRITE FILE ACTIVE? (BE45)
9E4C NOPE >>9E6C
9E4E YES, PRINTING A "#"?
9E50 NO >>9E56
9E52 YES, SAME AS PROMPT CHARACTER?
9E54 YES >>9E68
9E56 NO, PRINTING A RETURN CHAR?

9EB2 *********** APPLESOFT TRACE INTERCEPT **********************
(CONTROL PASSES HERE FOR EVERY STATEMENT)
(EXECUTED WHILE PRODOS IS ACTIVE)
9EB2 BUMP APPLESOFT LINE POINTER
9EB6 ---
9EBA MARK PROMPT FOR RECURSION
9EBC JUST IN CASE WE DIE IN HERE
9EBE RESTORE APPLESOFT'S STACK
9EC1 DOES BI KNOW WE ARE TRACING? (BE41)
9EC4 YES, REAL LIVE TRACE THEN >>9F39
BASIC Interpreter (BI) -- V1.1.1 -- 18 JUN 84
NEXT OBJECT ADDR: 9EC4

ADDR DESCRIPTION/CONTENTS

9EC6 ELSE, PICK UP NEXT TOKEN ON LINE
9ECA IS IT A TOKEN? >>9F81
9ECC OR END OF LINE? >>9EE5
9EEF NEITHER, DECREMENT STRING SPACECTR (BE49)
9ED1 OR >>9EEC
9ED3 COMPUTE SIZE OF FREE SPACE IN PGES
9ED7 AT LEAST 3 PAGES AVAILABLE?
9ED9 YES >>9EE5
9EEB NO, WRITE BUFFERED DATA <9F44>
9EE8 AND THEN GARBAGE COLLECT <9044>
9EE9 COMPUTE FREE SPACE NOW
9EFA AND SAVE IN STRING SPACECTR (BE49)
9EEA GET NEXT TOKEN
9EEC ---
9EEF JUMP BACK INTO APPLESOFT TO EXECUTE IT >>D820
9EF6 STORE TOKEN IN PROMPT
9EF8 LOOK UP TOKEN IN BI'S TOKEN TABLE (BE99)
9EF7 IT'S NOT ONE BI IS INTERESTED IN >>9EE5
9EF9 IT IS INTERESTING, CHANGE BRANCH (9EFF)
9EEC AND JUMP TO ONE OF THE FOLLOWING: >>9EEF

9EEF IF OR PRINT: PROMPT = 0
9F00 CLEAR OUT LAST CHAR SAVEAREA (BE4C)
9F03 GO TO MODE = C NEXT TIME THRU (BE83)
9F06 (BEGIN LOOKING FOR COMMANDS) (BE38)
9F0F NOW GO PROCESS THE IF OR PRINT >>9F2E

9F11 LIST: PROMPT = 0
9F13 (DONT LOOK FOR COMMANDS NOW)
9F15 GO DO IT >>9F2E

9F17 CALL: PROMPT = 1
9F19 (DONT LOOK FOR COMMANDS NOW)
9F1B GO DO IT >>9F2E

9F1D LET: DECREMENT STRINGCTR
9F1E AND GO BACK FOR NEXT TOKEN >>9ECE

9F21 TRACE: TURN TRACE ON (BE41)
9F24 THEN CONTINUE BELOW >>9F2A

9F26 NOTRACE: DROP INTO BACKGROUND TRACE (BE41)
9F29 CHANGE TOKEN TO "TRACE"
9F2A FORCE ON APPLESOFT TRACK
9F2B ---
9F2F GO BACK TO APPLESOFT TO PERFORM IT >>D820

BASIC Interpreter (BI) -- V1.1.1 -- 18 JUN 84
NEXT OBJECT ADDR: 9F2F

ADDR DESCRIPTION/CONTENTS

9F32 RESUME: CLEAR ONERR CODE
9F37 GO TO APPLESOFT TO PROCESS IT >>9ECE

9F39 RESTORE TRUE CSWL/KSWL <9A08>
9F3E PRINT "#" <FDED>
9F45 USE APPLESOFT TO PRINT CURRENT LINE NO. <ED24>
9F4A PRINT A BLANK SPACE <FDED>
9F4D PUT BI'S CSWL/KSWL INTERCEPTS BACK <9ABD>
9F51 THEN GO BACK AND HANDLE AS USUAL >>9E65

9F54 LOOKING FOR A LOWER CASE "c"
9F55 LOOKING FOR A "#"
9F5A STORE CHAR TO SEARCH FOR (9F61)
9F5B BRANCH BACK INTO APPLESOFT >>9E1E
9F60 BREAK IF Y IS ZERO111

9F61 "#" CHARACTER (ASOFT TRACE CHAR)

9F62 SAVE CALLER'S REGISTERS ***************
9F62 SAVE A,X AND Y REGS (BE3E)
9F6B RETURN

9F6C RESTORE CALLERS REGISTERS ***************
9F6C RESTORE A,X AND Y REGS (BE3E)
9F75 RETURN

9F76 SET MODE AND CSWL/KSWL ***************
9F76 STORE "STATE" MODE FROM X REGISTER (BE42)
9F7B COPY PROPER CSWL/KSWL VALUES TO REDIRECT... (BEF7)
9F7E VECTOR DEPENDING ON CURRENT MODE (BE38)
9F87 RETURN

9F88 PRINTERR: PRINT ERROR MSG ***************

9F88 ---
9F89 GET INDEX INTO PACKED MESSAGE TEXTS (BA13)
9F8C UNPACK MESSAGE INTO $281 <9FB0>
9F92 SAVE THE LENGTH (BC6E)
9F95 SKIP A LINE <9FAB>
9F9A PRINT A BELL <9FAD>
9F9D ---
9F9F PRINT CONTENTS OF $281 MSG BUFFER (0281)
9FAB PRINT A RETURN CHARACTER
9FAD AND EXIT >>FDED
BASIC Interpreter (BI) -- V1.1.1 -- 18 JUN 84  NEXT OBJECT ADDR: 9FAD

ADDR DESCRIPTION/CONTENTS

9FB0 ************** UNPACK ERROR MESSAGE ***************

9F80 NOTHING IN BUFFER AT FIRST
9FB6 GET A NIBBLE FROM PACKED MSG <9FD2>
9FB9 NON-ZERO, COMMON CHARACTER >>9FC0
9FB8 IF ZERO, GET NEXT NIBBLE <9FD2>
9FDE AND CONVERT TO UNCOMMON CHAR INDEX
9FC9 ---
9FC1 GET THE LETTER THIS NIBBLE REPRESENTS (BA48)
9FC4 ZERO? THEN END OF MESSAGE >>9FD1
9FC6 GET INDEX INTO OUTPUT BUFFER (BE48)
9FC9 AND STORE THE CHARACTER THERE (0201)
9FCC BUMP INDEX (BE48)
9FCF AND CONTINUE >>9FB6
9FD1 RETURN

9FD2 ************** UNPACK MESSAGE BYTE ***********************

9FD2 GET NEXT MSG BYTE (BA48)
9FD5 WORKING ON SECOND NIBBLE? >>9FDF
9FDF NO, TAB INDICATOR? >>9FDF
9FD9 NO, ISOLATE HIGH NIBBLE
9FDD NEXT TIME GET LOW NIBBLE
9DEE RETURN

9FDF ---

9FD8 GET TAB POSITION (BA48)
9F83 AND BUMP OUTPUT PTR ACCORDINGLY (BE4B)
9FEE THEN GO BACK FOR NEXT NIBBLE >>9FD2

9FE9 BUMP BYTE PTR FOR NEXT TIME
9FEA ISOLATE LOW NIBBLE
9FEE NEXT TIME GET HIGH NIBBLE
9FED RETURN

9FEE ************ WRITE ONE BUFFERED BYTE ******************

9FF0 SET UP COUNT OF 0801
9FF2 AND JUMP INTO ROUTINE BELOW >>A807

9FF4 ************ WRITE BUFFERED DATA/TEST ERROR ************

9FF4 WRITE BUFFERED DATA <A800>
9FF7 OK? THEN EXIT >>9F1C
9FFA ERROR, POP OUT OF THIS SUBROUTINE
9FFD AND GO TO ERROR HANDLER >>9AF9

BASIC Interpreter (BI) -- V1.1.1 -- 18 JUN 84  NEXT OBJECT ADDR: 9FFD

ADDR DESCRIPTION/CONTENTS

A000 ************ WRITE ALL BUFFERED DATA **********************

A000 ---
A002 GET BUFFERED DATA COUNT (BE4A)
A005 NONE BUFFERED? >>A018
A007 STORE BUFFERED DATA COUNT IN RW PARMS (BED9)
A00F MIL: WRITE <BE70>
A015 NOTHING BUFFERED NOW, COUNT=0 (BE4A)
A019 ERROR? >>A01C
A01B NO, EXIT
A01C RETURN

A01D ************ SPECIAL GARBAGE COLLECT *********************

(PULL OUT STRING CONSTANTS ALSO)

A01D DO GARBAGE COLLECTION NORMALLY FIRST <A044>
A020 ERROR? >>A43
A024 START OF STRING AREA = PROGRAM START PTR (BC84)
A026 USE GENERAL PURPOSE BUFFER (ABOVE HIMEM)
A028 FOR A GARBAGE COLLECT WORKAREA (BC7D)
A033 IT IS 3+1 PAGES IN LENGTH (BC7E)
A036 END OF STRING AREA IS AT END OF FREEAREA (BC86)
A040 GO COLLECT CONSTANT STRINGS NOW <A065>
A043 THEN EXIT

A044 ************ "FRE" COMMAND *****************************

(FAST APPLESOFTH STRING GARBAGE COLLECTION)

<p>| GENERAL PURPOSE BUFFR |
| (TOP OF OLD STRINGS) |
| HMEM --- |
| NEW STRINGS BUILDING |</p>
<table>
<thead>
<tr>
<th>DOWN</th>
</tr>
</thead>
<tbody>
<tr>
<td>'//_________________________________________________________________</td>
</tr>
<tr>
<td>OLD STRINGS</td>
</tr>
<tr>
<td>FREE AREA</td>
</tr>
</tbody>
</table>

TOP PART OF OLD STRINGS IS SAVED IN THE GENERAL PURPOSE BUFFER OR IN THE FREE AREA (WHICHEVER IS LARGER) AND A NEW COPY OF THE STRINGS IS BUILT JUST BELOW HIMEM.
A844 STRING AREA START IS ON PAGE BOUNDARY
A848B ASSUME 4 PAGE WORKAREA (BC7E)
A850 IN GENERAL PURPOSE BUFFER ABOVE HIMEM (BC7U)
A855 STRING START PTR IS START OF STRING AREA (BC84)
A859 COMPUTE NUMBER OF FREE PAGES
A85B AT LEAST ??
A85F IF NOT, USE G.P. WORKAREA INSTEAD >>A879
A85F DON'T USE ALL OF FREE AREA (LEAVE $300)
A861 NEW WORKAREA SIZE IS FREE AREA SIZE-$300 (BC7E)
A866 SET PTR TO WORKAREA AT FIRST FREE PAGE
A86D COMPUTE NUMBER OF STRING PAGES
A871 USE SMALLER OF STRING PAGES OR WORKAREA SIZE (BC7E)
A876 AS NEW WORKAREA SIZE (BC7E)
A879 END OF STRING AREA IS HIMEM
A885 RECORD WHETER LAST PAGE IS PARTIAL
A889 STRING START MSB IS HIMEM INITIALLY (BC96)
A88E ADJUST LORANGE AND HIRANGE MSB'S
A896 FOR PARTIAL PAGES AT EITHER END, (BC7F)
A893 SETTING THEM AT HIMEM FOR NOW.
A89C SET UP ARRAY END MSB +1 FOR COMPARES (BC82)
A89F $3E/$3F --> FIRST VARIABLE (LESS 7 BYTES)
A8A1 (EACH VARIABLE IS 7 BYTES)
A8AB SET UP ARRAY START LSB FOR COMPARES
A8B0 GET LORANGE VALUE (BC7F)
A8B4 PRIOR TO STRING AREA? (BC84)
A8B6 YES, THEN DONE >>A8F6
A8B8 ELSE, DROP LORANGE BY WORKAREA SIZE (BC7E)
A8BB AND SAVE THIS VALUE (BC7C)
A8BE NOW DROP IT ALSO BY THE DISTANCE BETWEEN
A8C0 ...THE OLD LORANGE AND THE STRING START PTR (BC7F)
A8CA USE THE LOWER OF THE TWO VALUES (BC7C)
A8CF TO PRODUCE THE MAXIMUM SIZED RANGE (BC7C)
A8D2 IS THIS BELOW THE BOTTOM OF THE STRINGS? (BC84)
A8D5 NO >>A8DC
A8D7 YES, USE THE BOTTOM POINTER INSTEAD (BC84)
A8DA (ADJUSTING FOR PARTIAL PAGE)
A8DC STORE FINAL LORANGE VALUE (BC7F)
A8DF COPY SOME PAGES BELOW HIRANGE TO WORKAREA <<A195
A8E2 (TO MAKE ROOM FOR NEW STRINGS)
A8E4 COLLECT SIMPLE STRING VARS FOR THIS RANGE <<A8F7
A8E7 ERROR? >>A8F4
A8E9 THEN COLLECT STRING ARRAYS <<A12D
A8EC NEW HIRANGE = OLD LORANGE (BC7F)
A8F2 CONTINUE LOOPING >>A89F
A8F4 IF ERROR, "RAM TOO LARGE" 
A8F6 EXIT TO CALLER

A8F7 ********** COLLECT SIMPLE STRINGS **********

A8F7 ---
A8F8 ADD 7 BYTES TO $3E/$3F PTR FOR NEXT VAR
A8F9 PTR ARRAYS NOW? 
A8FB IF SO, WE ARE DONE >>A12B
A8FC IS THIS A STRING VARIABLE?
A8FD NO >>A8F7
A8FE MAKE ABSOLUTELY SURE
A8FF GET MSB OF STRING POINTER
A903 IS IT WITHIN MY RANGE? (BC7F)
A906 NO >>A9F6
A909 NO >>A8F7
A910 YES, PULL IT OUT AND TACK IT TO HIMEM <<A1B8
A912 ALL WENT WELL, GET NEXT VARIABLE >>A9F8
A914 IF ERROR, EXIT NOW
A916 NORMAL EXIT TO CALLER
A91C RETURN

A92D ********** COLLECT STRING ARRAYS **********

A92D ---
A930 FIND THE NEXT ARRAY <<A15C
A933 NO MORE? >>A12B
A935 GOT ONE, GET MSB OF ITS STRING PTR
A938 WITHIN MY RANGE? (BC7F)
A93A NO >>A146
A93D NO >>A146
A93F YES, PULL IT OUT AND TACK IT TO HIMEM <<A1B8
A943 AND CONTINUE WITH NEXT ARRAY ELEMENT >>A147
A945 ERROR EXIT
A946 ---
A94A BUMP POINTER TO NEXT ARRAY MEMBER
A94A POINTER NOW AT NEXT ARRAY? (BC61)
A950 NO, DO THIS ELEMENT >>A132
A952 NO >>A132
A955 YES, SET UP TO PROCESS THAT ONE THEN >>A12D
A95C ---
A95D $3E --> ARRAY VARIABLES (BC61)
A9561 END OF ARRAY VARS
A9566 NO, CONTINUE >>A16C
A956A YES, OUT (CARRY SET, NO MORE ARRAYS) >>A194
BASIC Interpreter (BI) -- V1.1.1 -- 18 JUN 84

ADDRESSES

A16C POINT TO ARRAY FOLLOWING THIS (LSB AND...)  
A176 MSB TO X REGISTER  
A17D CHECK TYPE OF VARIABLE  
A182 SKIP INTEGER AND REAL ARRAYS >>A15C  
A186 GET NUMBER OF DIMENSIONS  
A188 *2 TO SKIP SIZES  
A189 +5 TO SKIP FIXED STUFF AT BEGINNING  
A190 POINT TO FIRST ARRAY MEMBER  
A191 READY TO ROLL, $3E POINTS TO IT  
A194 RETURN  

A195 ********** COPY PAGES TO WORKAREA **********************  
TO MAKE ROOM FOR NEW STRINGS BEING MOVED  
TO HIMEM, COPY SOME STRING PAGES FROM OLD  
STRING AREA TO THE WORKAREA TO PROTECT THEM.  

A195 $3A/$3B --- FIRST PAGE TO SAVE (BC7C)  
A199 $3C/$3D --- WORKAREA (BC7D)  
A1A5 COPY N+1 PAGES (SIZE OF WORKAREA) (BC7B)  
A1A9 ---  
A1B7 EXIT WHEN FINISHED  

A1B8 ********** PULL STRING OUT ********************************  
TACK STRING JUST UNDER HIMEM AT CURRENT  
STRING START POINTER.  

A1BB IS STRING BELOW SAVED AREA? (BC7C)  
A1BD YES, IT'S STILL THERE THEN >>A1C4  
A1C4 $3A/$3B --- STRING  
A1CF DROP STRING START PTR BY LEN OF THIS STRING  
A1D0 UPDATE STRING'S LSB IN VARIABLE PTR  
A1D8 FIX UP MSB OF STRING START PTR ALSO  
A1DD AND OF VARIABLE PTR  
A1E1 IS THIS A NULL LENGTH STRING?  
A1E3 YES, NO MOVE TO DO >>A1EE  
A1EE ---  
A1E7 ELSE, COPY STRING OUT  
A1EE ---  
A1EF OUT OF FREESPACE? (BC82)  
A1F4 RETURN TO CALLER WITH INDICATION  

A1F5 ********** ALLOCATE BUFFER ***********************  
A1F5 NEED 4 PAGES

********** GENERAL PURPOSE ALLOCATE **********  
A1F7 STORE THAT (BB47)  
A1FA GO GARBAGE COLLECT TO GETスペース <A044>  
A1FD ERROR? >>A24A  
A201 HOW MANY FREE PAGES ARE THERE?  
A203 ARE THERE ENOUGH? (BB47)  
A206 IF NOT, "RAM TOO LARGE" MSG  
A208 TOO FEW... >>A24A  
A28A GOT ENOUGH, $3A--TOP OF FREESPACE  
A211 AND $3C--NEW TOP AFTER ALLOCATION  
A21B COMPUTE LENGTH OF STRINGS FOR COPY  
A229 COPY STRINGS DOWN "N" PAGES IN MEMORY <A35B>  
A22F SUBTRACT "N" FROM STRING ADDRESS MSB'S (BB47)  
A235 ADJUST ALL POINTERS IN SIMPLE & ARRAY VARS <A39F>  
A23A OLD HIMEM BECOMES BUFF ADDR HIGH WATER MARK (BB49)  
A241 NEW HIMEM IS "N" PAGES LOWER  
A246 FIND PAGE JUST BEYOND A FILE BUFFER (BC88)  
A249 RETURN  
A24A ---  
A24B RETURN  

A24C ********** FREE BUFFER ******************************  
A24C GARBAGE COLLECT STRINGS <A044>  
A24E ERROR? >>A299  
A255 PUT HIMEM-$100 INTO $3A/3B  
A259 AND HIMEM+$400 INTO $3C/3D  
A25F (COPY LSB'S)  
A266 BC92 = LENGTH OF STRINGS (BC92)  
A270 COPY STRINGS UP 4 PAGES <A37F>  
A275 PREPARE TO ADJUST THEM BY $400 (BC87)  
A27B NEW HIMEM+$400  
A27D ADJUST ALL STRING ADDRS UP BY $400 <A39F>  
A283 ARE WE FREEING BOTTOM-MOST BUFFER?  
A285 YES, DONE >>A2B3  
A288 CHECK OPEN FILE COUNT (BB4D)  
A2B8 NONE OPEN? (HOW CAN THAT BE?) >>A297  
A28D WHICH FILE'S BUFFER IS NEXT TO HIMEM?  
A292 SEARCH UNTIL IT IS FOUND... >>A29A  
A297 ---  
A299 RETURN IF NO FILE IS USING THIS BUFFER  
A29A ---  
A29B GIVE THAT FILE THE BUFFER PASSED TO US (BC99)  
A29E (SURE HOPE THAT FILE WAS FLUSHED!) (BC93)  
A29A PASS FILE REF NUM 'O'S MLI (BC7)  
A2AE MLI: SET NEW BUFFER <BE70>  
A2B1 ERROR? >>A299  
A2B3 ---  
A2B4 RETURN
BASIC Interpreter (BI) -- V1.1.1 -- 18 JUN 84  NEXT OBJECT ADDR: A2B4

A2B5 ********** GETBUF: GET A BUFFER **********************

THIS ROUTINE IS CALLED THROUGH AN EXTERNAL ENTRY POINT IN THE GLOBAL PAGE. IT ALLO- CATES A FIXED LOCATION BUFFER BETWEEN THE BI AND ITS BUFFERS.

A2B5 ALLOCATE A BUFFER OF ANY SIZE (A=PAGE) <A1F7>
A2B6 ERROR? >>A300
A2B7 FIND FIRST PAGE OF BUFFER (BB4A)
A2C4 GET FILE OPEN COUNT (BB4D)
A2C7 NONE OPEN? >>A2EA
A2CA BUFFER PAGE PTR BY $400 (BB49)
A2CD TO POINT TO PREVIOUSLY ALLOCATED
A2CF BUFFER. (BB49)
A2D2 FIND OPEN FILE WITH THIS BUFFER (BC93)
A2D7 GET IT, (BE9C)
A2DA SET FILE BUFFER REAL LOW IN MEMORY <A352>
A2DD THEN SET IT TO NEW BUFFER LOCATION <A29B>
A2EB BELOW ALL OTHERS (BEC9)
A2ED DO THIS FOR EACH OPEN FILE...
A2ED THEREBY INSERTING A BLANK BUFFER >>A2D2
A2ED IS EXEC FILE ACTIVE? (BE43)
A2F0 NO, DONE >>A2FF
A2F2 YES, A2F4 MOVE EXEC BUFFER DOWN ALSO <A352>
A2FD AND BUMP UP ABOVE IT
A2FF EXIT TO CALLER
A300 RETURN

A301 ********** FREEBUF: FREE BUFFER **********************

THIS ROUTINE IS CALLED THROUGH AN EXTERNAL ENTRY POINT IN THE GLOBAL PAGE. IT FREES A FIXED LOCATION BUFFER PREVIOUSLY ALLO- CATED BY GETBUF.

A301 GET COUNT OF OPEN FILES (BE4D)
A305 INDEX THIS BY 4 PAGES PER FILE
A306 ADD TO HIMEM MSB
A308 SAVE THIS AS TOP OF BUFFERS (BB49)
A30D THEN SET UP BOTTOM AS HIMEM MSB (BB4A)
A310 GET OLD ORIGINAL HIMEM (BEFORE ANY BUFFERS) (BEFB)
A313 SAME AS THIS ONE?
A315 THEN NOTHING ELSE TO DO >>A358
A317 ASSUME NO BUFFERS BY REPLACING OLD HIMEM
A319 ANY EXEC FILE OPEN? (BE43)
A31C NO, CONTINUE >>A323
A31E YES, MOVE EXEC BUFFER TO OLD HIMEM <A2F2>
A321 AND GO MOVE HIMEM DOWN BY $400 >>A341
A323 ELSE, START WITH TOP BUFFER (BB49)
A326 ANY OPEN FILES? (BE4D)

BASIC Interpreter (BI) -- V1.1.1 -- 18 JUN 84  NEXT OBJECT ADDR: A329

A329 IF NOT, WE ARE DONE >>A34D
A32B SEARCH FOR OPEN FILE WITH THIS BUFFER (BC93)
A32E NOT IT? >>A34A
A330 G0 IT, GIVE IT NEW HOME AT HIMEM
A332 AND SET BUFFER LOW <A352>
A335 THEN TO NEW LOC <A29B>
A339 DROP TOP BUFFER PTR BY $400 (BB49)
A341 AND DROP HIMEM BY $400
A346 AND GO DO NEXT BUFFER >>A323
A34A ---
A34B (LOOP TO SEARCH FOR OPEN FILES) >>A32B
A34D WHEN FINISHED, GARBAGE COLLECT <A044>
A350 ---
A351 THEN EXIT NORMALLY TO CALLER

********** SET BUFFER BELOW ALL OTHERS ***

A352 ---
A353 USE BOTTOM BUFFER PTR (BB4A)
A356 SET FILE BUFFER <A29B>
A35A AND EXIT

A35B ********** COPY BLOCK DOWN IN MEMORY ***************

A35B COPY ALL FULL PAGES DOWN TO THEIR NEW HOME
A362 COPYING $3a-->$3c
A369 BUMP BOTH MSB'S
A36D DROP PAGE COUNTER (BC93)
A370 AND CONTINUE >>A362
A372 NO SHORT LAST PAGE? (BC92)
A375 THEN EXIT NOW >>A37E
A377 ELSE, COPY PARTIAL PAGE
A37E THEN EXIT

A37F ********** COPY BLOCK UP IN MEMORY ***************

A37F PARTIAL PAGE? (BC92)
A382 NO, JUST COPY FULL PAGES NOW >>A38B
A384 YES, COPY SHORT PAGE FIRST <A396>
A387 DROP BOTH MSB'S
A388 PAGE COUNT GONE TO ZERO? (BC93)
A38E YES, DONE >>A39E
A390 ELSE, DROP PAGE COUNT (BC93)
A393 AND GO COPY A FULL PAGE UP >>A384
A396 ---
A397 COPY REMAINDER OF PAGE UP (BACKWARDS)
A39E RETURN
BASIC Interpreter (BI) -- V1.1.1 -- 18 JUN 84  NEXT OBJECT ADDR: A39E

A39F **************** ADJUST ALL STRING ADDRES ***********************
(BC87 HAS ADDITIVE ADJUSTMENT FACTOR)

A39F USE LOMEM PAGE AS MSB FOR $3E/3F
A3A3 GET LOMEM LSB
A3A5 AND END OF SIMPLE VARS PAGE
A3A8 JUMP INTO THE LOOP >>A3AF
A3AA ---
A3AB SKIP ONE SIMPLE VARIABLE
A3AF ---
A3B1 OVERFLOW? >>A3B5
A3B3 YES, BUMP MSB
A3B5 FINISHED WITH SIMPLE VARS?
A3B9 (CHECK BOTH MSB AND LSB OF PTR)
A3BB ---
A3BC YES... >>A3D2
A3BE NO,
A3C0 LOOK AT A SIMPLE VARIABLE
A3C5 SKIP INTEGER AND REAL VARS >>A3AA
A3C7 (DOUBLE CHECK MSB)
A3CB ITS A STRING, POINT TO ITS LEN/ADD
A3CC ADJUST IT IF NECESSARY <A3F9>
A3CF THEN SKIP OVER IT >>A3AA

A3D2 COPY ARRAYS STARTING LSB
A3D4 (MSB IS IN X REGISTER NOW) (BC91)
A3D7 ---
A3DB FIND A STRING ARRAY <A15C>
A3D8 NO MORE? THEN DONE... >>A40C
A3DD ---
A3E0 ADJUST ITS ADDRESS IF NEED BE <A3F9>
A3E5 SKIP TO NEXT STRING ELEMENT OF ARRAY
A3EE AT END OF THIS ARRAY YET? (BC91)
A3F1 NO... >>A3DD
A3F3 (CHECK MSB ALSO)
A3FF YES... GO GET NEXT ARRAY >>A3D7

A3F9 **************** ADJUST A STRING ADDRESS ***********************

A3F9 GET STRING LENGTH
A3FB IGNORE NULL STRINGS >>A40C
A3FD POINT TO MSB OF ADDRESS
A3FF IS STRING STORED OUTSIDE OF PROGRAM?
A403 NO, LEAVE IT ALONE >>A40C
A405 STORE ABOVE LOMEM, ADD FACTOR TO MSB
A40C THEN EXIT

BASIC Interpreter (BI) -- V1.1.1 -- 18 JUN 84  NEXT OBJECT ADDR: A40C

A40D **************** COMPRESS ALL ASOFT VARS ***********************
THIS ROUTINE SQUASHES ALL APPLESOFT VARS
UP AGAINST THE BOTTOM OF THE STRINGS

A40D | STRINGS |
A410 | ARRAY VARS |
A418 | SIMPLE VARS |

A40D GARBAGE COLLECT FIRST <A1D1>
A410 ERROR? >>A471
A412 COMPUTE LENGTH OF SIMPLE AND ARRAY VARS
A417 AND SAVE IT (BC89)
A427 NEXT, COMPUTE LENGTH OF SIMPLE VARS ONLY
A42B AND SAVE IT (BC8B)
A435 SUBTRACT VAR LENGTH FROM STRING START
A437 TO FIND A PLACE TO PUT THE VARS UNDER (BC92)
A43A THE STRINGS (START ON AN EVEN PAGE BOUND)
A440 $3C/$3D --- PLACE TO PUT VARS
A447 $3A/$3B --- START OF VARS (ROUNDED TO EVEN
A449 PAGE ALIGNEMENT)
A44F COPY VARS UP AGAINST STRINGS <A37F>
A454 STORE START OF VARS PTR (BC8E)
A457 SUMPING PAGE NUMBER BY ONE
A463 SUBTRACT THIS PTR FROM HOMEM TO COMPUTE (BC90)
A466 TOTAL LENGTH OF COMBINED VARS/STRINGS
A468 AND SAVE THIS TOO (BC8D)
A46B ALSO, SAVE HOMEM MSB IN CASE THEY ARE MOVED
A471 DONE, EXIT

A472 ************ REEXPAND COMPRESSED VARS *************
THIS ROUTINE MOVES SIMPLE AND ARRAY VARS
BACK DOWN TO LOMEM.

A472 | STRINGS |
|
A472 | FREE SPACE |

BASIC Interpreter (B1) -- V1.1.1 -- 18 JUN 84  NEXT OBJECT ADDR: A5D4

ADDR     DESCRIPTION/CONTENTS
-------------------------
A5D4    10 OR MORE HOURS (TWO DIGITS?)
A5D7    IN ANY CASE, CONVERT HOURS <A62F>
A5DB    IF TWO DIGITS ... >>A5DE
A5DE    ---
A5E2    CONVERT YEAR (LEFT ZERO FILL) <A60A>
A5E6    GET MONTH INDEX (*9) (BCB3)
A5E9    POINT TO LAST CHARACTER
A5EC    COPY MONTH NAME FROM TABLE (B9BD)
A5EF    TO LINE (0201)
A5F7    BACKWARDS ... >>A5EC
A5FB    PUT A "-" IN (0201)
A5FE    TWO PLACES (0205)
A607    EXIT BY CONVERTING DAY >>A62F

A60A *********** CONVERT 2 DIGIT NUMBER ***********
             (FORCE LEFT ZERO FILL)

A60A    ---
A60B    ADD 100 TO FORCE SIGNIFICANCE IN TENS
A60D    CONVERT IT <A62F>
A610    IGNORE 100'S PLACE
A611    RETURN

A612 *********** CONVERT TO HEX ***********

A612    ---
A613    ISOLATE LOW NIBBLE
A615    AND GO CONVERT IT FIRST <A61D>
A619    NOW ISOLATE HIGH NIBBLE
A61C    AND FALL THRU TO CONVERT IT ALSO

A61D    CONVERT NIBBLE TO NUMERIC ASCII
A61F    97
A621    NO >>A625
A623    YES, CONVERT SBA-SBF TO SC1-SC6
A625    AND STORE THE RESULT (0201)
A628    BUMP LINE INDEX BACK
A629    PRECEED WITH A $ SIGN
A62E    RETURN

A62F *************** CONVERT TO DECIMAL *******************

A62F    A,X = NUMBER Y=INDEX TO LAST FIELD DIGIT (BCB9)
A632    STORE NUMBER IN ACCUMULATOR (BCB9)
A635    DIVIDE BY 10 <A640>
A638    GET DIGIT AND CONVERT IT (BCB2)
A63D    STORE IN LINE (0201)
A640    AND DROP LINE INDEX BY ONE
A641    IS QUOTIENT NOW ZERO? (BCB9)
A64A    NO, CONTINUE UNTIL IT IS >>A635
A64C    ELSE, EXIT

************ DIVIDE ACCUMULATOR BY 10 ************
A64D    24 BIT SHIFT (3 BYTES)
A651    CLEAR SUM (BCB2)
A654    INDEX ACCUMULATOR LEFT ONE BIT <A6D7>
A657    ALSO ROLL 4TH BYTE OF ACCUM (BCB2)
A65B    IF MSB = 10... (BCB2)
A665    THEN ADD ONE TO ACCUMULATIVE SUM (BCAP)
A66B    ---
A669    SHIFT 24 TIMES >>A654
A66B    RETURN
A66C    ---
A676    RETURN

A677 ************* SYNTAX: PARSE COMMAND LINE *************
             (ALSO EXTERNAL ENTRY FOR COMMAND STRINGS)
A677    INIT COMMAND NUMBER TO -1
A67E    A BLANK ENDS EACH STRING (BCA9)
A683    AT MOST 8 CHARACTERS IN A COMMAND (BCAA)
A686    PARSE COMMAND ITSELF <A1A8>
A689    GET FIRST LETTER (BCBD)
A68C    MUST BE ALPHABETIC
A68E    IT IS... >>A697
A690    IT'S NOT, IS IT A "-"?
A692    YES, OK THEN... >>A697
A694    ELSE, ITS BAD - SYNTAX ERROR >>A639
A697    SCAN FOR COMMAND IN TABLES <A6E1>
A699    BAD COMMAND? >>A694
A69C    NO, IMMEDIATE COMMAND MODE? (BE42)
A69F    NO, DEFERRED... >>A6AC
A6A1    IMMEDIATE, EXEC ACTIVE? (BE43)
A6A4    YES, NEVER MIND >>A6AC
A6A6    ERASE TO END OF LINE <FC9C>
A6A9    AND GO TO A NEW LINE ON SCREEN <9FAB>
A6AC    ASSUME NO PARMS AT ALL
A6A4    NO PATH NAME YET (BCBD)
A6A7    NO SECONDARY PATH NAME EITHER (0200)
A6BD    CURRENT SLOT = DEFAULT SLOT (BE61)
A6C3    CURRENT DRIVE = DEFAULT DRIVE (BE62)
A6CB    BUFFER ALLOCATION = HIMEM (BCB8)
A6CC    GET LENGTH OF COMMAND NAME (BE52)
A6D0    ALLOW 2 MORE CHARACTERS FOR NOW (BCAA)
A6D3    ARE ANY PARAMETERS PERMITTED? (BE54)
A6D4    NO... MUST BE MON OR NONMON >>A736
A6DB    YES (02#) PR#7
A6D9    YES... >>A739
A6DB    ELSE, REPARSE THE COMMAND <A1A8>
A6D9    FOR THIS COMMAND... (BE54)
A6E3 DOES THE PREFIX NEED FETCHING? >>A6EA
A6E5 YES,
A6E7 ML1: GET PREFIX FROM DEFAULT DRIVE <BE70>
A6EA ---
A6EC END OF LINE? >>A736
A6EE NO, COMMA?
A6F0 NO >>A6F5
A6F2 YES, NO FILENAME, LOOK FOR KEYWORDS >>A787
A6F5 "/"?
A6F7 YES >>A6FD
A6F9 NO, ALPHABETIC?
A6FB NO... FILE NAMES MUST BEGIN THAT WAY >>A72F
A6FD ---
A6FE DON'T FLUSH ANY BLANKS OUT OF PATHNAME
A703 ALLOW 64 CHARACTERS NEXT PARSE
A709 PARSE NEXT OPERAND ON LINE <A1F>
A70D SAVE ITS LENGTH (BCBC)
A712 FOUND A PATHNAME\#1 (BE56)
A715 COPY PARM KEYWORD TO $288 (BCBC)
A719 (ASSUMING PATHNAME1-PATHNAME2) (BE28)
A71F CHECK NEXT CHAR (OTHER THAN A BLANK) <AA3A>
A722 NOT COMMA OR RETURN, BAD1 >>A72C
A724 RETURN? >>A798
A726 NO, PATHNAME EXPECTED NOW? (BE54)
A72A YES, ALL IS WELL >>A762
A72C NO, "SYNTAX ERROR" >>A839
A72F NON ALPHA FILE NAME, CHECK COMMAND NUMBER (BE53)
A731 IS IT "RUN"?
A734 NO, ERROR >>A72C
A736 YES, ITS OK THEN (MIGHT BE "RUN 100") >>A798
A739 IN#/$R$S, REPARSE COMMAND <AA1B>
A73C RETURN FOUND - ERROR >>A72C
A73E "A"? (ADDRESS KEYWORD)
A740 IF SO, GO PARSE THAT KEYWORD ONLY >>A78C
A742 ELSE, ZERO ACCUMULATOR <AB37>
A745 CONVERTING ONE BYTE'S WORTH (BCAC)
A74A PUT IT IN PR#/$IN# SLOT VALUE AREA (BCAC)
A74F FOUND SLOT FOR PR#/$IN# (BE56)
A752 CONVERT SLOT # <A968>
A755 ERROR? >>A761
A757 GET CONVERTED VALUE (BE60)
A75A >>8?
A75C NO, ITS OK >>A791
A75E YES, "RANGE ERROR"
A761 RETURN
A762 SECOND PATHNAME EXPECTED?
A763 NO >>A787
A765 YES, FLUSH TO NON-BLANK <AA3A>
A768 NOTHING ELSE ON LINE???? >>A72C
A76B DON'T FLUSH ANY BLANKS OUT OF PATHNAME
A772 COPY SECOND PATHNAME TO $281 <AA80>
BASIC Interpreter (BI) -- V1.1.1 -- 18 JUN 84  NEXT OBJECT ADDR: A810

ADDR DESCRIPTION/CONTENTS

A810 EXTERNAL COMMAND? IF SO GO NOW! >>A836
A812 MY OWN COMMAND, "PREFIX"?
A814 YES, GO NOW >>A836
A818 S OR D VALID KEYWORDS FOR THIS CMD?
A81B NO, GO NOW >>A836
A820 PATHNAME1 GIVEN WITH THIS COMMAND?
A821 NO, GO NOW >>A836
A823 YES, GET FILE INFO FOR PATHNAME1 <87D8>
A826 NO ERRORS I HOPE >>A836
A828 ERROR WAS PATH NOT FOUND?
A82A NO, REAL ERROR = SAY SO >>A83B
A82F CAN WE CREATE PATHNAME1?
A831 YES, OK THEN >>A836
A833 ELSE, "PATH NOT FOUND"
A835 RETURN
A836 GO TO COMMAND HANDLING ROUTINE >>BCAB

A839 ********** SYNTAX ERROR ********************

A839 LOAD BI CODE FOR "SYNTAX ERROR"
A83B AND RETURN WITH ERROR CONDITION
A83C RETURN

A83D ********** ADD PREFIX TO PATHNAMES **********

A83D GET SLOT NUMBER (BE61)
A844 PUT SLOT IN HIGH 3 BITS
A848 ADD DRIVE TO TOP BIT AND SHIFT SLOT DOWN (BE62)
A84B ...TO FORM THE UNIT NUMBER (BEC7)
A853 READ THE PATHNAME PREFIX TO $201 (BEC9)
A85D MLI: ONLINE <BE70>
A868 ERROR? >>A83B
A86D DEFAULT DRIVE = PARSED DRIVE (BE3D)
A86F DEFAULT SLOT = PARSED SLOT (B3C)
A871 PATHNAME1 STARTS WITH "/"?
A873 THEN IT'S ALREADY GOT A PREFIX >>A866
A876 ELSE, GET LENGTH OF PATHNAME
A87A BUMP IT BY 2 (TO ALLOW FOR '/S')
A882 WITH PREFIX WILL IT EXCEED 64 CHAR?
A887 YES, "SYNTAX ERROR" >>A877
A889 NO; UPDATE LENGTH TO INCLUDE PREFIX (BCBC)
A88F ---
A893 AND COPY PATHNAME1 FORWARD TO MAKE ROOM (BCBD)
A89C PUT A "/'" AT THE BEGINNING
A8A1 AND AT THE END (BCBD)
A8A4 COPY PREFIX JUST READ TO START OF PATHNAME1 ($200)
A8AA GET COMMAND NUMBER (BE53)
A8AD "OPEN"?
A8AF YES, DONE NOW! >>A866
A8B1 YES, "APPEND"?
A8B3 YES, DONE NOW! >>A866

---

BASIC Interpreter (BI) -- V1.1.1 -- 18 JUN 84  NEXT OBJECT ADDR: A8B5

ADDR DESCRIPTION/CONTENTS

A8B5 "EXEC"?
A8B7 YES, DONE NOW! >>A866
A8B9 ELSE, GET LENGTH OF PATHNAME2 (BE80)
A8BB COMBINE THIS WITH PREFIX LENGTH (BE01)
A8BC MORE THAN 64 CHAR?
A8C6 IF SO, "SYNTAX ERROR" >>A8B7
A8CB UPDATE LENGTH (BE80)
A8CB COPY PATHNAME2 FORWARD TO MAKE ROOM (BE81)
A8DB PUT A "/'" IN FIRST
A8DD THEN THE PREFIX AND ANOTHER SLASH (BE81)
A8DE ---
A8EF DONE!

A8EB ********** KEYWORD LOOKUP *******************

A8BE ZERO THE ACCUMULATOR <AB37>
A8B8 NINE POSSIBLE KEYWORDS IN TABLE
A8BD COMPARE AGAINST EACH (B968)
A8BF FOUND IT? >>A927
A8F5 NO, IS IT "T"? (FILE TYPE)
A8F7 YES, OK THEN >>A8FC
A8F9 ELSE, BAD KEYWORD >>A839
A8FC IT'S "T", IS IT PERMITTED ON THIS CMD?
A901 NO, ERROR >>A923
A906 ELSE, MARK WE HAVE "T" (BE56)
A90B START WITH TYPE INDEX OF 0 (BCAD)
A910 INDICATE WHERE T VALUE IS TO GO (BCE4)
A913 AND GO PASE ONE CHAR <AA3A>
A916 NOTHING THERE?? >>A8F9
A918 IS IT A $?
A91A YES, HE GAVE TYPE IN HEX >>A976
A91C IS IT ALPHABETIC?
A91E NO, CONVERT DECIMAL TYPE >>A960
A920 ELSE, GO LOOKUP TYPE NAME IN TABLE >>A9B6
A923 ---
A924 "INVALID PARAMETER"
A926 RETURN

A927 GET BIT POSITION OF THIS KEYWORD (B975)
A92A IGNORE "V" >>A947
A92C IS THIS KEYWORD PERMITTED? (BE55)
A92F NO, NOT WITH THIS COMMAND ANYWAY >>A923
A931 S OR D?
A933 NO >>A941
A935 YES, ALREADY FOUND IT ON THIS LINE? (BE57)
A93B YES, DON'T CHANGE DRIVE DEFAULT >>A947
A93A ELSE, ASSUME DRIVE = 1
A941 MARK WE HAVE SLOT/DRIVE (BE57)
A947 GET SIZE-1 IN BYTES OF VALUE (B97F)
BASIC Interpreter (BI) -- V1.1.1 -- 18 JUN 84  NEXT OBJECT ADDR: A954

ADDR  DESCRIPTION/CONTENTS

A954  AND OFFSET TO VALUE IN STORAGE AREA (B9AE)
A957  FLUSH TO NON-BLANK (<AA3>)
A95A  NOTHING ELSE THERE? >>A9B0
A95C  IS NEXT CHAR A "$"?
A95E  YES, GO CONVERT HEX - ELSE, FALL THRU >>A976

A960  ********** CONVERT DECIMAL NUMBER ***********************
A964  SAVE LINE INDEX (BE4B)
A966  CONVERT/ADD ONE DECIMAL DIGIT TO ACCUM (<AA5C)
A966  OK... >>A96C
A968  OVERFLOW? THEN "RANGE ERROR" >>A9B3
A96A  BAD DIGIT? THEN "SYNTAX ERROR" >>A9B8
A96C  RESTORE LINE INDEX (BE4B)
A96F  FLUSH TO NEXT NON-BLANK (<AA3>)
A972  AND GO BACK TO CONVERT NEXT DIGIT >>A968
A974  ALL DONE, END OF LINE OR COMMA >>A98F

A976  ********** CONVERT HEX NUMBER ***********************
A979  PLUSH TO NEXT NON-BLANK (SKIP "$") (<AA3A)
A980  NOTHING LEFT? >>A9B0
A984  BAD DIGIT? THEN "RANGE ERROR" >>A9B3
A986  OVERFLOW? THEN "RANGE ERROR" >>A9B0
A987  RESTORE LINE INDEX (BE4B)
A98A  FLUSH TO NEXT NON-BLANK (<AA3A)
A98D  AND GO CONVERT NEXT DIGIT >>A97B

A98F  ********** STORE KEYWORD VALUE ***********************
A996  IF NUMBER IS A SHORT INTEGER >>A9B3
A94A  COPY ACCUM TO PROPER PARM STORAGE CELL (BCAF)
A93B  RESTORE LINE INDEX (BE4B)
A93F  AND EXIT

A986  "SYNTAX ERROR" JUMP >>A939
A983  "RANGE ERROR" JUMP >>A75E

A9B6  ********** STORE KEYWORD VALUE ***********************
A9B6  ---
A9B6  COPY 3 CHARACTER TYPE TO ACCUM (BCAF)
A9B6  (COPIED ALL 3?) >>A9C7
A9C0  (GET NEXT CHAR IGNORING BLANKS) (<AA3A)
A9C5  MUST HAVE 3 CHARACTERS! >>A980

BASIC Interpreter (BI) -- V1.1.1 -- 18 JUN 84  NEXT OBJECT ADDR: A9C7

ADDR  DESCRIPTION/CONTENTS

A9C7  SAVE LINE INDEX (BE4B)
A9CA  INITIALIZE NAME INDEX TO ZERO
A9CF  HAVE ALL 13 BEEN CHECKED?
A9D1  YES, NO MATCH >>A9B0
A9D4  ELSE, INDEX\*3 (BCAD)
A9D9  COMPAR TYPE GIVEN (BCAF)
A9DB  TO TYPES IN TABLE (B997)
A9DE  (IGNORE MSB'S)
A9DF  NO MATCH ALREADY... >>A9E9
A9E3  ELSE,
A9E5  CHECK ALL THREE CHAR >>A9D9
A9E7  THEY ALL MATCH! WE FOUND IT >>A9EE
A9E9  NOT THE RIGHT ONE, (BCAD)
A9EC  GO TRY THE NEXT ONE >>A9CA
A9EE  REVERSE NAME INDEX
A9F5  AND GET TYPE VALUE FROM TABLE (B989)
A9FB  STORE IT IN TYPE VALUE STORAGE AREA (B96A)
A9FF  RESTORE LINE INDEX (BE4B)
A9F6  AND EXIT

AA00  ********** COPY PATHNAME2 ***********************
AA00  GET NEXT CHARACTER (<AA4A)
AA03  HEX HD STORE IT INDEXED OFF $200 ($200)
AA07  COMMA?
AA09  YES, DONE >>A37
AA0B  BLANK?
AA0D  YES, DONE >>A37
AA0F  RETURN?
AA11  YES, OUT NOW >>AA46
AA13  PATHNAME TOO LONG? (BCAA)
AA16  NO, CONTINUE COPYING >>AA00
AA18  ELSE, SET NOT-EQUAL CONDITION
AA1A  AND EXIT

AA1B  ********** COPY COMMAND NAME INTO TXTBUF ***********************
AA1B  SET INDICES
AA1F  GET NEXT NON-BLANK (<AA4A)
AA22  COPY TO TXTBUF (BCBD)
AA26  COMMA?
AA28  YES, DONE >>A37
AA2A  BLANK?
AA2C  YES, DONE >>A37
AA2E  RETURN?
AA30  YES, DONE >>AA48
AA32  AT MAX LENGTH (8)? (BCAA)
AA35  NO, CONTINUE >>AA1F
AA37  ELSE, SET NOT-EQUAL CONDITION
AA39  AND EXIT
BASIC Interpreter (BI) -- V1.1.1 -- 18 JUN 84 NEXT OBJECT ADDR: AA39

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>AA3A</td>
<td>*************** FLUSH TO NON-BLANK ***********************</td>
</tr>
<tr>
<td></td>
<td>Z-FLAG SET IF COMMA OR RETURN FOUND</td>
</tr>
<tr>
<td></td>
<td>C-FLAG SET IF COMMA</td>
</tr>
<tr>
<td>AA3A</td>
<td>IGNORE BLANKS</td>
</tr>
<tr>
<td>AA3F</td>
<td>GET NEXT NON-BLANK &lt;AA4A&gt;</td>
</tr>
<tr>
<td>AA42</td>
<td>COMMA?</td>
</tr>
<tr>
<td>AA44</td>
<td>YES, OUT &gt;&gt;AA49</td>
</tr>
<tr>
<td>AA46</td>
<td>RETURN?</td>
</tr>
<tr>
<td>AA48</td>
<td>EXIT INDICATING WHAT WE FOUND</td>
</tr>
<tr>
<td>AA49</td>
<td>RETURN</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>AA4A</td>
<td>*************** GET NEXT CHARACTER ***********************</td>
</tr>
<tr>
<td></td>
<td>AA4A GET NEXT CHAR IN INPUT LINE (0200)</td>
</tr>
<tr>
<td></td>
<td>AA4D FORCE ZERO MSB</td>
</tr>
<tr>
<td></td>
<td>AA4F LOWER CASE?</td>
</tr>
<tr>
<td>AA51</td>
<td>NO &gt;&gt;AA55</td>
</tr>
<tr>
<td>AA53</td>
<td>YES, FORCE UPPER CASE</td>
</tr>
<tr>
<td>AA55</td>
<td>BUMP LINE INDEX</td>
</tr>
<tr>
<td>AA56</td>
<td>IS THIS A FLUSH CHARACTER (LIKE BLANK)? (BC9)</td>
</tr>
<tr>
<td>AA59</td>
<td>YES, GO GET NEXT ONE &gt;&gt;AA4A</td>
</tr>
<tr>
<td>AA5B</td>
<td>ELSE, RETURN WITH IT</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>AA5C</td>
<td>*************** CONVERT DIGIT AND ADD TO ACCUM ***********************</td>
</tr>
<tr>
<td>AA5E</td>
<td>NUMERIC?</td>
</tr>
<tr>
<td>AA5E</td>
<td>NO &gt;&gt;AA64</td>
</tr>
<tr>
<td>AA62</td>
<td>YES &gt;&gt;AA65</td>
</tr>
<tr>
<td>AA64</td>
<td>NOT NUMERIC, EXIT WITH CARRY SET</td>
</tr>
<tr>
<td>AA65</td>
<td>AND Z-FLAG RESET</td>
</tr>
<tr>
<td>AA67</td>
<td>RETURN</td>
</tr>
<tr>
<td>AA68</td>
<td>ISOLATE DECIMAL PORTION OF DIGIT</td>
</tr>
<tr>
<td>AA6B</td>
<td>CURRENT VALUE OF ACCUM... (BC81)</td>
</tr>
<tr>
<td>AA6E</td>
<td>&gt;1,703,936?</td>
</tr>
<tr>
<td>AA70</td>
<td>YES, OVERFLOW &gt;&gt;AA94</td>
</tr>
<tr>
<td>AA74</td>
<td>PUSH ENTIRE ACCUM ONTO STACK (BCAF)</td>
</tr>
<tr>
<td>AA7B</td>
<td>ACCUM*2 (ROL IT ONCE) &lt;AAD7&gt;</td>
</tr>
<tr>
<td>AA7E</td>
<td>ACCUM*4 (AND AGAIN) &lt;AAD7&gt;</td>
</tr>
<tr>
<td>AA84</td>
<td>---</td>
</tr>
<tr>
<td>AA85</td>
<td>ACCUM<em>4+ACCUM &lt;&lt; ACCUM</em>5 (BCAF)</td>
</tr>
<tr>
<td>AA91</td>
<td>FINALLY, ACCUM<em>5+2 --&gt; ACCUM</em>10 &lt;AAD7&gt;</td>
</tr>
<tr>
<td>AA94</td>
<td>---</td>
</tr>
<tr>
<td>AA95</td>
<td>ACCUM OVERFLOW? &gt;&gt;AA9A</td>
</tr>
<tr>
<td>AA97</td>
<td>NO, ADD NEW DIGIT TO ACCUM (BCAF)</td>
</tr>
<tr>
<td>AA9A</td>
<td>AND STORE IT (BCAF)</td>
</tr>
<tr>
<td>AA9D</td>
<td>NO CARRY? &gt;&gt;AAAD</td>
</tr>
<tr>
<td>AA80</td>
<td>GOT CARRY, PROPAGATE IT THRU ACCUM (BC80)</td>
</tr>
<tr>
<td>AAAA</td>
<td>OVERFLOW ERROR</td>
</tr>
<tr>
<td>AAAD</td>
<td>NORMAL EXIT</td>
</tr>
</tbody>
</table>
BASIC Interpreter (BI)  --  V1.1.1  --  18 JUN 84  NEXT OBJECT ADDR: AB24

AB25 NOT THE ONE, SKIP TO NEXT (BE52)
AB3E IF THERE ARE ANY MORE >>AAFA
AB30 ELSE, NO SUCH COMMAND (BE53)
AB34 RETURN THRU $BE06 VECTOR >>BE06

AB37 ************ ZERO THREE BYTE ACCUM ***********************
AB37 ZERO THE THREE BYTE WORK
AB39 ...ACCUMULATOR (BCAF)
AB42 RETURN

AB43 *********** "=" COMMAND ***********************
AB43 CHECK FILE TYPE (BE88)
AB46 APPLESOFT PROGRAM?
AB48 YES, "RUN" IT >>AB2
AB4A BINARY FILE?
AB4C YES, "BRUN" IT >>AB0D
AB4E TEXT FILE?
AB50 NO >>AB55
AB52 YES, "EXEC" IT >>B221
AB55 SYS FILE?
AB57 YES, GO RUN IT >>AB5D
AB59 ELSE, "FILE TYPE MISMATCH"
AB5C RETURN

****** RUN "SYS" FILE *************
AB5D CLOSE ALL OPEN FILES <B4F2>
AB60 CLOSE EXEC <B2FB>
AB65 LSB OF A$ IS $0 (BE58)
AB68 FREE UP ALL OF BI'S MEMORY (BF6B)
AB78 A$2000 IS WHERE IT WILL LOAD (BE59)
AB88 TYPE IS "SYS" (BE6A)
AB8A FORCE, T, PATHNAME1, AD PARMS (BE56)
AB8D GO DO A STANDARD BRUN >>AE16

AB98 ************* "CHAIN" COMMAND *************
AB99 SQUASH VARIABLES UP AGAINST HIMEM <A40D>
AB95 SAVE HIMEM (BC7B)
AB9C SET NEW HIMEM BELOW COMBINED VARS
AB9E LOAD FILE (LEAVE OTHERS OPEN) <AC03>
ABA4 RESTORE OLD HIMEM
ABA6 ERROR? >>AC14
ABAB NO, CLEAR VARIABLES <D665>
ABAB REEXPAND VARIABLES <D665>
ABB0 THEN GO "RUN" PROGRAM >>ABC7

ABB2 ************ "RUN" COMMAND ***********************
ABB2 NO INPUT FILE ACTIVE NOW
ABB7 NO APPLESOFT ERROR NUMBER
ABBC GOT PATHNAME1?
ABBD NO, ERROR >>ABD5
ABBF YES, LOAD PROGRAM <ABFE>
ABCD ERROR? >>AC14
ABCA NO, CLEAR VARIABLES <D665>

ABCC CLEAR ERROR FLAG
ABCD POSITION TO LINE NUMBER IF GIVEN <AC97>
ABCE RESTORE MY INTERCEPTS <A8BD>
ABCF CLEAR COMMAND NUMBER ETC., MODE = 4 <ABD5>
ABD2 JUMP INTO APPLESOFT TO RUN PROGRAM >>D7D2

ABD5 *********** CLEAR COMMAND NUMBER ETC. *************
ABD5 SET NORMAL (NON-INVERSE OR FLASH) <F273>
ABDA SEARCH CHARACTER FOR TRACE IS "#" (9F61)
ABDF NO COMMAND NUMBER NOW (BE53)
ABE2 NO PROMPT
ABE6 SET MODE=4 (DEFERRED) <9F76>
ABE9 "SYNTAX ERROR" IF THINGS GO WRONG >>AB39

ABEC *********** "LOAD" COMMAND *************
ABEC LOAD PROGRAM <ABFE>
ABEF ERROR? IF NOT, FALL THRU TO WARMSTART >>AC14

ABF1 ************ WARMDO$: WARMSTART BI *************
ABF1 CLEAR APPLESOFT, RESET POINTERS <D665>
ABF4 RESET MODE/SFT INTERCEPTS <9A17>
ABF9 CURSOR HORIZ. = ø (START OF LINE)
ABFB GO WARMSTART APPLESOFT >>D43F

ABFE ************ LOAD A PROGRAM *************
ABFE CLOSE ALL OPEN FILES <B4F2>
AC01 ERROR? >>AC14
AC03 GO LOAD FILE <AC15>
AC06 ERROR? >>AC14
AC08 SET LOMEM = ARRAYS = FREESTART
AC0A ALL TO END OF PROGRAM LOADED
AC14 RETURN
AC15  ******************** READ A PROGRAM FROM A FILE ********************

AC15  READ REQUESTED
AC17  TYPE = BAS ASSUMED
AC19  OPEN THE FILE <B194>
AC1C  ERROR?  >>AC14
AC20  MLI: GET EGF <B7E8>
AC23  ERROR?  >>AC14
AC27  APPLESOFT PROGRAM START ==> READ DATA (BED7)
AC2A  ADD TO THAT THE EOF MARK TO ... (BEC8)
AC2D  SET AD PARM ==> END OF PROGRAM IMAGE (BE58)
AC3B  OVERFLOW?  >>AC1F
AC3D  NO, WOULD PROGRAM EXCEED HIMEM?
AC3F  IF SO...
AC41  "PROGRAM TOO LARGE"  >>AC14
AC43  ELSE, PICK UP LENGTH AGAIN (BEC8)
AC49  AND GO READ IT IN <AF89>
AC4C  ERROR?  >>AC14
AC4B  CLOSE FILE <AF94>
AC51  ERROR?  >>AC14
AC53  RELOCATE PROGRAM IF NECESSARY <AC61>
AC5C  COPY AD PARM TO APPLESOFT PGM END PTR
AC68  RETURN

AC61  ******************** RELOCATE APPLESOFT PROGRAM********************

AC61  ---
AC62  RKS APPLESOFT PROGRAM SAVED FROM SAME
AC64  MEMORY LOCATION?  (BE99)
AC73  YES, NOTHING TO DO THEN  >>ACBA
AC79  ELSE, LOOP THROUGH PROGRAM
AC78  ADJUSTING ALL ADDRESSES TO
AC7D  THE NEW LOAD LOCATION

AC97  ******************** POSITION TO LINE NUMBER ********************

AC97  WAS A LINE NUMBER PARM GIVEN?  (BE57)
AC9D  NO, NEVER MIND >>ACBA
AC9F  COPY L KEYWORD VALUE TO APPLESOFT'S LINE # (BE68)
ACA9  THEN CALL APPLESOFT TO FIND THE LINE <D61A>
ACAF  SUBTRACT ONE FROM THE ADDRESS
ACB1  AND POINT APPLESOFT'S GETCHR SUBROUTINE
ACB3  AT IT (SO NEXT CHAR READ WILL BE FIRST
ACB5  CHARACTER ON THE LINE).
ACBA  RETURN

ACBB  ******************** "SAVE" COMMAND ********************

ACBB  DOES FILE EXIST ALREADY?  >>ACDF
ACBD  NO, TYPE = BAS
ACBF  IN T KEYWORD VALUE (BE6A)
ACC2  AND MLI LIST (BE88)
ACCC  ALLOW ALL ACCESSES (READ/WRITE/ETC.) (BE77)
ACCG  SAVE PROGRAM START ADDRESS IN (BEA5)
ACCF  AUXID'S (BE99)
ACDA  GO CREATE A NEW FILE <AD46
ACDD  ERROR?  >>AD28

ACDF  WRITE ACCESS REQUESTED
ACE1  BAS TYPE FILE
ACE3  OPEN IT <B194>
ACE6  ERROR?  >>AD28
ACEB  SUBTRACT APPLESOFT PTRS TO COMPUTE
ACED  LENGTH OF PROGRAM.
ACEB  STORE THIS IN EOF MARK LIST (BE88)
ACB7  MSG OF EOF MARK IS 80 (<E64 PGM) (BECA)
ACBF  POINT LIST TO PROGRAM AS DATA TO WRITE (BED7)
ACBD  WRITE A RANGE TO DISK FILE <AF9C>
ACBE  ERROR?  >>AD28
ACBF  MLI: SET EOF (TO TRUNCATE OLD LONGER FILE) <B7E0>
ACD2  ERROR?  >>AD28
ACD4  CLOSE THE FILE <AF94>
ACD7  ERROR?  >>AD28
ACD8  DOES PROGRAM START MATCH AUXID IN FILE INFO?
ACD0  NO, CHANGE IT >>AD29
ACD8  ELSE, EXIT
ACD9  TO CHANGE IT, (BE99)
ACD2  EXIT THRU SET FILE INFO ROUTINE >>B7D9

AD32  ******************** "CREATE" COMMAND ********************

AD32  AUXID = 0 (A$ OR RECLN)
AD3D  TYPE KEYWORD GIVEN?
AD3F  YES >>AD46
AD43  NO, ASSUME TYPE = DIR (BE6A)
AD46  *** CREATE FILE ENTRY *** (BE43)
AD49  EXEC FILE ACTIVE?
AD4C  HOW MANY FILES ARE OPEN INCLUDING EXEC?  (BE4D)
AD4F  8 OR MORE?
AD51  YES, ERROR >>AD6E
AD56  ELSE, SET TYPE IN MLI LIST (BE4A)
AD59  FULL ACCESS (READ/WRITE/ETC.)
AD5B  KIND = STANDARD FILE
AD5D  DIR FILE WANTED?
BASIC Interpreter (BI) -- V1.1.1 -- 18 JUN 84 NEXT OBJECT ADDR: AD5F
AD5F  NO >>AD63
AD61  YES, KIND = DIR FILE
AD63  SET ACCESS (BEA3)
AD66  AND KIND (BEA7)
AD68  MLl: CREATE (DON'T COME BACK HERE) >>BE70
AD6E  "RAM TOO LARGE" ERROR
AD70  RETURN
AD71  ********** "RENAME" COMMAND **************
AD71  ---
AD75  SECOND PATHNAME GIVEN?
AD78  IF SO, GO MLl: RENAME >>AD7F
AD7A  "SYNTAX ERROR" OTHERWISE >>A839
AD7D  *********** "DELETE" COMMAND ***************
AD7D  SETUP MLl: DELETE CALL TYPE
AD7F  EXIT THRU MLl CALL >>BE70
AD82  *********** "LOCK" COMMAND *******************
AD82  GET FILE INFO FOR PATHNAME1 <B7D0>
AD85  GET ACCESS CODES (BEA7)
AD88  TURN OFF ALL...
AD8A  BUT READ
AD8F  THEN GO SET UPDATED FILE INFO >>B7E7
AD92  *********** "UNLOCK" COMMAND *******************
AD92  GET FILE INFO FOR PATHNAME1 <B7D0>
AD95  TURN ON ALL FILE ACCESSES
AD9D  THEN GO SET UPDATED FILE INFO >>B7E7
ADA0  *********** "PREFIX" COMMAND ******************
ADA0  SLOT/DRIVE GIVEN ON COMMAND? (BE57)
ADA6  IF SO, GOT OPERAND ALREADY >>ADAC
ADA8  ELSE, (BE56)
ADA8  CHECK FOR PATHNAME1
ADA9  AND GO DO MLl: SET PREFIX ...
ADAE  IF IT'S THERE >>AD7F
ADB0  ELSE, IS BASIC PROGRAM RUNNING?
ADB2  IF SO, SET PREFIX ACTIVE FLAG >>ADDI
ADB4  NO, NEW LINE <9FAB>
ADBC  END OF NAME YET? >>AD99
ADBE  NO, COPY NAME IN PATHNAME1 BUFFER (BCBD)
ADC3  TO OUTPUT DEVICE <9FAD>
ADC9  AND SKIP A BLANK LINE <9FAB>
ADD0  DONE

BASIC Interpreter (BI) -- V1.1.1 -- 18 JUN 84 NEXT OBJECT ADDR: ADD0
ADDA  ********** "SAVE" COMMAND **********
ADD1  SET PREFIX ACTIVE FLAG
ADD3  SO BASIC CAN READ THE PREFIX (BE46)
ADD7  RETURN
ADD8  ********** "SAVE" COMMAND **********
ADD8  PATHNAME1 FOUND? >>AE0E
ADD9  NO, NEW FILE (BE57)
ADD0  AD, L, AND E POSSIBLE
ADDF  AD AND EITHER L OR E REQUIRED
ADD1  OR ELSE ERROR >>AE12
ADD6  PUT AD IN CREATE PARAMETER LIST (BEA5)
ADD9  AND IN GET FILE INFO LIST (BE99)
ADPF  TYPE = BIN ASSUMED (BE6A)
AE00  T KEYWORD GIVEN?
AE02  IF SO, ERROR >>AE12
AE04  GO CREATE THE FILE <AD46>
AE07  ERROR? >>AE14
AE09  GET FILE INFO <B7D0>
AE0C  ERROR? >>AE14
AE0E  WRITING...
AE10  GO PROCESS LIKE A LOAD OTHERWIS >>AE25
AE12  "PATH NOT FOUND" ERROR
AE14  ---
AE15  RETURN
AE16  ********** "RUN" COMMAND **********
AE16  (DOES NOT SET MODE=4 SO DOS COMMANDS MAY
AE19  NOT BE ISSUED AS WITH A BASIC PROGRAM)
AE16  BLOAD IT FIRST <AE23>
AE19  ERROR? >>AE14
AE1B  THEN CALL IT <AE28>
AE1E  THEN EXIT
AE1F  RETURN
AE20  INDIRECT JMP TO BINARY PROGRAM >>BED7
AE23  ********** "LOAD" COMMAND **********
AE23  READING...
AE25  TYPE = BIN
AE27  OPEN THE FILE <B194>
AE2A  ERROR? >>AE14
AE2C  ASSUME USER SPECIFIED AD KEYWORD (BE58)
AE35  IF SO, USE HIS ADDRESS >>AE47
AE37  ELSE, USE AD IN FILE INFO AUXID (BE99)
AE40  T KEYWORD GIVEN?
AE42  YES, INVALID NAME (ONLY BIN IS LEGAL >>AE78
AE47  POINT READ/WRITE FARMS TO DATA (BED7)
BASIC Interpreter (BI) -- V1.1.1 -- 18 JUN 84  NEXT OBJECT ADDR: AE4D

AE4D PICK UP LENGTH FROM L KEYWORD VALUE (BE5F)
AE53 WAS L OR E GIVEN?
AE55 NEITHER >>AE7C
AE57 BOTH?
AE59 YES...NAUGHTY! >>AE78
AE5B E GIVEN?
AE5D NO, MUST BE L >>AE92
AE5F YES... (BE5D)
AE63 COMPUTE L = (E - AD) (BE58)
AE6F PLUS ONE FOR INCLUSIVE RANGE >>AE72
AE72 MAKE SURE NO BORROW OCCURRED >>AE92
AE74 OR ELSE, "RANGE ERROR"
AE77 RETURN

AE78 "INVALID PARM" ERROR
AE7B RETURN

AE7C ---

AE7E MLI: GET EOF <BE70>
AE81 ERROR? >>AE90
AE83 GET L (EOF MARK) (BECA)
AE89 BETTER NOT EXCEED 64K (BECA)
AE8C NO... >>AE92

AE9E YES, "PROGRAM TOO LARGE"
AE9F ---

AE91 RETURN

AE92 STORE LENGTH TO READ OR WRITE (BE9D)
AE9B B KEYWORD GIVEN?
AE9D NO >>AEC4

AAE1 YES, COPY B VALUE TO SET MARK LIST (BE5A)
AAEA ---

AAE3 MLI: SET MARK <BE70>
AAEB NO ERROR? >>AEC4

AAE4 ERROR, RANGE ERROR?

AAE6 NO >>AE90

AAE8 BSAVING (NOT BLOAD/BRUNING)?

AAEA NO >>AE90

AAEB MLI: FORCE EOF FORWARD TO MARK <BE70>

AAEC AND TRY SET MARK AGAIN >>AEEA

AAE3 RETURN

AAE4 GET COMMAND NUMBER (BE53)

AAE7 ASSUME READ

AAE9 BSAVE?

AAEC NO, READ IS CORRECT >>AECD

AAED WRITING

AAEF MLI: READ OR WRITE <BE70>

AED2 ERROR? >>AE90

AED4 THEN EXIT THRU CLOSE >>AP94

BASIC Interpreter (BI) -- V1.1.1 -- 18 JUN 84  NEXT OBJECT ADDR: AE4D

AED7 ********** "STORE" COMMAND **********

AED7 PATHNAME EXISTS? >>AEEB

AEE9 NO, T = VAR BY DEFAULT

AEE1 FULL ACCESS (READ/WRITE/ETC.)

AEE6 CREATE THE FILE <AD46>

AEE9 ERROR? >>AF39

AEE8 COMPRESS APPLESOFT VARS AGAINST HIMEM <A40D>

AEP4 OPEN "VAR" FILE FOR WRITE <B194>

AEP7 ERROR? >>AF32

AEP9 POINT TO INTERNAL 5 BYTE HEADER BUFFER <A3A>

AEPF AND WRITE OUT LENGTHS OF VARS <AF9C>

AEPF ERROR? >>AF32

AF01 STORE ADDRESS OF VARS (BC8E)

AF04 IN READ/WRITE PARM LIST (BED7)

AF07 AND FILE INFO AUXID (BB9B)

AF13 GET LENGTH OF VARS (BC91)

AF19 AND WRITE THEM OUT <AF9C>

AF1C ERROR? >>AF32

AF20 MLI: GET MARK <BE78>

AF25 MLI: SET NEW EOF (TRUNCATE IF NECESSARY) <BE78>

AF28 ERROR? >>AF32

AF2A SET FILE INFO WITH AD OF VARS <B7D9>

AF2D ERROR? >>AF32

AF2F CLOSE FILE <AF94>

AF32 ---

AF34 REEXPAND VARS BACK AGAIN <A472>

AF39 RETURN

AF3A ********** 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

AF3A STORE ADDRESS OF 5 BYTE INFO

AF3C IN READ/WRITE PARM LIST (BED7)

AP46 LENGTH = 5

AF48 RETURN

AF49 ********** "RESTORE" COMMAND **********

AF49 TYPE = VAR

AF48 READING

AF4D OPEN THE FILE <B194>

AF50 ERROR? >>AF39

AF52 SET UP TO READ THE HEADER <AF3A>

AF55 READ 5 BYTE HEADER <AF9B>

AF58 ERROR? >>AF39

AF5A PICK UP WHERE TO READ IN COMPRESSED VARS (BE9B)
<table>
<thead>
<tr>
<th>BASIC Interpreter (BI) -- V1.1.1 -- 18 JUN 84</th>
<th>NEXT OBJECT ADDR: AF5D</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADDR  DESCRIPTION/CONTENTS</td>
<td></td>
</tr>
<tr>
<td>AF5D  FROM AUIDX (BC8E)</td>
<td></td>
</tr>
<tr>
<td>AF63  ADJUST MSB OF THIS BY THE DIFFERENCE</td>
<td></td>
</tr>
<tr>
<td>AF66  BETWEEN HIMEM'S (NOW AND WHEN STORED) (BC8D)</td>
<td></td>
</tr>
<tr>
<td>AF73  MAKE SURE VARS WONT OVERLAY PROGRAM</td>
<td></td>
</tr>
<tr>
<td>AF75  IF SO, ERROR  &gt;&gt;AF96</td>
<td></td>
</tr>
<tr>
<td>AF77  COMPUTE LENGTH OF ALL VARS/STRINGS</td>
<td></td>
</tr>
<tr>
<td>AF81  (HIMEM-START) (BCBF)</td>
<td></td>
</tr>
<tr>
<td>AF85  GO READ COMBINED VARS INTO MEMORY  &gt;&gt;AF99</td>
<td></td>
</tr>
<tr>
<td>AF88  ERROR?  &gt;&gt;AF99</td>
<td></td>
</tr>
<tr>
<td>AF8A  CLOSE THE FILE  &gt;&gt;AF94</td>
<td></td>
</tr>
<tr>
<td>AF8D  EXIT BY REEXPANDING THE VARS DOWN &gt;&gt;AF32</td>
<td></td>
</tr>
<tr>
<td>AF90  &quot;PROGRAM TOO LARGE&quot; ERROR</td>
<td></td>
</tr>
<tr>
<td>AF93  RETURN</td>
<td></td>
</tr>
<tr>
<td>AF94  ********** CLOSE FILE ***************</td>
<td></td>
</tr>
<tr>
<td>AF94  SET MLI CLOSE OPCODE</td>
<td></td>
</tr>
<tr>
<td>AF96  AND GO TO MLI &gt;&gt;AF4</td>
<td></td>
</tr>
<tr>
<td>AF98  ********** READ/WRITE A RANGE **********</td>
<td></td>
</tr>
<tr>
<td>AF98  READ MLI OPCODE</td>
<td></td>
</tr>
<tr>
<td>AF9A  JUMP IN &gt;&gt;AF9E</td>
<td></td>
</tr>
<tr>
<td>AF9C  WRITE MLI OPCODE</td>
<td></td>
</tr>
<tr>
<td>AF9E  STORE LENGTH (BEDA)</td>
<td></td>
</tr>
<tr>
<td>AFA4  EXIT THRU MLI:READ OR WRITE &gt;&gt;BE76</td>
<td></td>
</tr>
<tr>
<td>AFA7  ********** &quot;PR#&quot; COMMAND *****************</td>
<td></td>
</tr>
<tr>
<td>AFA7  USE CSWL AND OUTVEC</td>
<td></td>
</tr>
<tr>
<td>AFA7  JUMP TO COMMON CODE &gt;&gt;AF9</td>
<td></td>
</tr>
<tr>
<td>AFAE  ********** &quot;IN#&quot; COMMAND *****************</td>
<td></td>
</tr>
<tr>
<td>AFAE  USE KSWL</td>
<td></td>
</tr>
<tr>
<td>AFBD  AND INVEC</td>
<td></td>
</tr>
<tr>
<td>AFBB  *2 FOR USE AS INDEX INTO TABLE</td>
<td></td>
</tr>
<tr>
<td>AFBD  WAS SLOT PARAMETER GIVEN?</td>
<td></td>
</tr>
<tr>
<td>AFBF  NO...  &gt;&gt;AFD2</td>
<td></td>
</tr>
<tr>
<td>AFCA  YES,  (BE57)</td>
<td></td>
</tr>
<tr>
<td>AFCA  AD GIVEN?  &gt;&gt;AF7</td>
<td></td>
</tr>
<tr>
<td>AFCE  NO, GET INVEC OR OUTVEC FOR THIS SLOT (BE10)</td>
<td></td>
</tr>
<tr>
<td>AFCE  NO AND STORE ON AD KEYWORD VALUE (BE56)</td>
<td></td>
</tr>
<tr>
<td>AFDF  VALIDITY CHECK I/O DRIVER &lt;&lt;AF9&gt;</td>
<td></td>
</tr>
<tr>
<td>AFDF  NO GOOD?        &gt;&gt;AF6</td>
<td></td>
</tr>
<tr>
<td>AFD7  GET INDEX TO CSWL OR KSWL (BCA9)</td>
<td></td>
</tr>
<tr>
<td>AFDF  AND REPLACE ONE OR THE OTHER WITH (B036)</td>
<td></td>
</tr>
<tr>
<td>AFDF  HIS ADDRESS (BE59)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>BASIC Interpreter (BI) -- V1.1.1 -- 18 JUN 84</th>
<th>NEXT OBJECT ADDR: AF66</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADDR  DESCRIPTION/CONTENTS</td>
<td></td>
</tr>
<tr>
<td>AF66  RETURN</td>
<td></td>
</tr>
<tr>
<td>AF67  VALIDITY CHECK AD KEYWORD VALUE  &gt;&gt;AF99</td>
<td></td>
</tr>
<tr>
<td>AF68  NO GOOD?  &gt;&gt;AF6</td>
<td></td>
</tr>
<tr>
<td>AF69  EJECT OR CPW VALUE TO INVEC OR OUTVEC (BE59)</td>
<td></td>
</tr>
<tr>
<td>AF69  EXIT BUT DON'T REDIRECT I/O NOW</td>
<td></td>
</tr>
<tr>
<td>AF69  ********** VALIDITY CHECK I/O DRIVER *******</td>
<td></td>
</tr>
<tr>
<td>AF69  $3A/3B --&gt; NEW HANDLER (FROM AD PARM) (BE58)</td>
<td></td>
</tr>
<tr>
<td>B005  IS DRIVER IN MAIN RAM (BELOW $C000)?</td>
<td></td>
</tr>
<tr>
<td>B007  YES  &gt;&gt;B01E</td>
<td></td>
</tr>
<tr>
<td>B009  NO, RESET I/O CARD ROMS (CFPF)</td>
<td></td>
</tr>
<tr>
<td>B00C  USE $3C TO COUNT ITERATIONS</td>
<td></td>
</tr>
<tr>
<td>B00E  TEST ROM AT USER'S ADDRESS</td>
<td></td>
</tr>
<tr>
<td>B014  FOR STABILITY</td>
<td></td>
</tr>
<tr>
<td>B018  256 TIMES</td>
<td></td>
</tr>
<tr>
<td>B01C  MUST BE OK</td>
<td></td>
</tr>
<tr>
<td>B01D  RETURN</td>
<td></td>
</tr>
<tr>
<td>B01E  MAIN I/O DRIVER</td>
<td></td>
</tr>
<tr>
<td>B020  MUST START WITH &quot;CLD&quot; INSTRUCTION</td>
<td></td>
</tr>
<tr>
<td>B022  OK...  &gt;&gt;B01C</td>
<td></td>
</tr>
<tr>
<td>B024  ELSE, &quot;NO Device Connected&quot;</td>
<td></td>
</tr>
<tr>
<td>B027  RETURN</td>
<td></td>
</tr>
<tr>
<td>B028  ********** &quot;BYE&quot; COMMAND *******</td>
<td></td>
</tr>
<tr>
<td>B028  CLOSE ANY OPEN FILES  &gt;&gt;B4F2</td>
<td></td>
</tr>
<tr>
<td>B029  CLOSE EXEC FILE, IF ANY  &gt;&gt;B2FB</td>
<td></td>
</tr>
<tr>
<td>B030  MLI CALL;  &gt;&gt;B0F0</td>
<td></td>
</tr>
<tr>
<td>B033  QUIT</td>
<td></td>
</tr>
<tr>
<td>B034  USE READ PARMLIST BECAUSE QUIT DOESN'T NEED PARMS.</td>
<td></td>
</tr>
<tr>
<td>B036  ********** &quot;CAT&quot; COMMAND *******</td>
<td></td>
</tr>
<tr>
<td>B036  39 CHARACTERS PER LINE</td>
<td></td>
</tr>
<tr>
<td>B038  THEN PROCESS LIKE &quot;CATALOG&quot; &gt;&gt;B03C</td>
<td></td>
</tr>
<tr>
<td>B03A  ********** &quot;CATALOG&quot; COMMAND *******</td>
<td></td>
</tr>
<tr>
<td>B03A  79 CHARACTERS PER LINE</td>
<td></td>
</tr>
<tr>
<td>B03C  STORE LINE LENGTH (BCB6)</td>
<td></td>
</tr>
<tr>
<td>B042  TEST FOR T AND</td>
<td></td>
</tr>
<tr>
<td>B044  pathnamei GIVEN</td>
<td></td>
</tr>
<tr>
<td>B045  GOTO T &gt;&gt;B04A</td>
<td></td>
</tr>
<tr>
<td>B047  NO T, T=0 (ANY TYPE WILL DO) (BE6A)</td>
<td></td>
</tr>
<tr>
<td>B04A  GIV PATHNAMEI &gt;&gt;B051</td>
<td></td>
</tr>
<tr>
<td>B04C  NO PATHNAMEI, GET FILE INFO FOR PREFIX  &lt;B7D0&gt;</td>
<td></td>
</tr>
<tr>
<td>B04F  ERRONE &gt;&gt;B097</td>
<td></td>
</tr>
<tr>
<td>B051  OPEN/READ DIRECTORY HEADER &lt;B14A&gt;</td>
<td></td>
</tr>
</tbody>
</table>
BASIC Interpreter (B1) -- V1.1.1 -- 18 JUN 84

ADDRESS DESCRIPTION/CONTENTS

B054  ERROR? >>B0B7
B056  SKIP TO A NEW LINE <9FAB>
B059  FORMAT DIRECTORY'S NAME TO $201 <B0B8>
B05C  PRINT $201 <9F9D>
B05F  SKIP TO A NEW LINE <9FAB>
B062  BLANK $201 BUFFER <->66C>
B067  UNPACK HEADING MESSAGE LINE <9FB0>
B06A  PRINT IT (40 OR 80 COLUMNS) <9F9D>
B06D  SKIP TO A NEW LINE <9FAB>
B073  ANY FILES IN THIS DIRECTORY? (BCBA)
B076  NO >>B0A3
B078  YES, READ NEXT ENTRY <B1D1>
B07B  ERROR? >>B0B7
B07D  GET TYPE REQUESTED FOR SEARCH (B66A)
B080  ANY TYPE WILL DO? >>B077
B082  NO, CHECK TYPE AGAINST THIS ENTRY (0267)
B085  NOT IT, SKIP IT >>B08D
B087  ELSE, FORMAT ENTRY TO $201 <A4C4>
B09A  AND PRINT $201 <9F9D>
B09D  CHECK KEYBOARD (C000)
B09F  FOR A CONTROL-C
B092  IGNORE ANYTHING ELSE >>B09E
B094  CONTROL-C, WHAT STATE ARE WE IN? (BE42)
B097  DEFERRED >>B0A3
B099  NO, IMMEDIATE, RESET KEYBOARD STROBE (C010)
B09C  AND EXIT RIGHT NOW >>B0A3
B09E  ELSE, ANY FILES LEFT IN COUNT? (BCBA)
B0A1  YES, CONTINUE >>B078
B0A3  ELSE, CLOSE DIRECTORY <AF94>
B0A6  ERROR? >>B0B7
B0A8  SKIP TO A NEW LINE <9FAB>
B0AB  FORMAT BLOCKS FREE AND IN USE TO $201 <B0E7>
B0AE  ERROR? >>B0B7
B0B0  PRINT $201 <9F9D>
B0B3  SKIP A LINE <9FAB>
B0B7  DONE

B0B8  ********** FORMAT NAME OF DIRECTORY **********

B0B8  BLANK $201 BUFFER <->66C>
B0BB  FILE NAME IS AT $1 INTO DIR ENTRY
B0BD  GET NAME LENGTH/TYPE (025D)
B0C2  VOLUME DIRECTORY HEADER?
B0C4  NO >>B0C9
B0C6  YES, START NAME WITH "/" (0200)
B0CA  ----
B0CB  ISOLATE NAME LENGTH FROM TYPE
B0CD  AND SET UP LENGTH TO COPY (0200)
B0D2  COPY DIRECTORY NAME TO (0259)

B0E7  ********** FORMAT BLOCKS FREE/INUSE **********

B0E7  POINT MLI:ONLINE PARMLIST
B0E9  TO TXTBUF (PATHNAME1) (BEC9)
B0F1  COPY DEVICE NUMBER (UNIT) (BF30)
B0F9  MLI: ONLINE <BE78>
B0FC  ERROR? >>B0B7
B101  ISOLATE NAME LENGTH FROM BUFFER
B104  BUMP BY ONE TO INCLUDE "/"
B105  AND STORE IT AS A PREFIX (BC8C)
B10A  STORE "/" AS FIRST CHARACTER (BCBD)
B10D  GET FILE INFO FOR PREFIX <B700>
B110  ERROR? >>B0B7
B112  BLANK $201 BUFFER <->66C>
B117  UNPACK "BLOCKS FREE; BLOCKS USED.." <9FB0>
B11A  ZERO THE THREE BYTE ACCUM <AB37>
B125  CONVERT AUXID (TOTAL BLOCKS) <->62F2
B138  CONVERT BLOCKS USED <->62F2
B137  BLOCKS FREE = TOTAL BLOCKS (BEBC)
B13E  ... = BLOCKS USED (BE8D)
B145  CONVERT BLOCKS FREE <->62F2
B149  DONE1

B14A  ********** OPEN/READ DIRECTORY HDR **********

B14A  READ ONLY
B14E  CHECK FILE KIND (BE8B)
B151  VOLUME DIRECTORY?
B153  NO >>B158
B155  YES, TYPE = DIR (BE88)
B158  OPEN THE FILE <B1A8>
B15B  ERROR? IF NOT, FALL THRU >>B193

B15D  ********** READ DIRECTORY HDR **********

B15D  BUFFER IS $259
B169  LENGTH IS $2B (ONE ENTRY) (BED9)
B173  MLI: READ <BE78>
B176  ERROR? >>B193
B17A  COPY ENTRY LENGTH, ENTRIES PER BLOCK, (027C)
B17D  AND FILE COUNT FROM DIR HDR (BEC7)
B183  STORE ENTRY LENGTH IN READ LENGTH ROW (BED9)
B188  SET COUNTER TO FIRST ENTRY IN BLOCK (BC8B)
B18D  MARK = 0 (START OF FILE) (BEC9)
B193  RETURN
**BASIC Interpreter (BI) -- V1.1.1 -- 18 JUN 84 NEXT OBJECT ADDR: B193**

**ADDR** | **DESCRIPTION/CONTENTS**
--- | ---
B194 | ************ OPEN FILE ******************************
A REGISTER = ACCESS BITS
X REGISTER = DEFAULT TYPE
B194 | ---
B198 | T KEYWORD GIVEN?
B19A | NO >>B19F
B19C | YES, USE KEYWORD VALUE INSTEAD (BE6A)
B19F | ---
B1A0 | EXISTING FILE OF THIS TYPE? (BE88)
B1A3 | NO, ERROR >>B1C9
B1A5 | CHECK ACCESS REQUESTED (BE78)
B1A8 | REQUESTED ACCESS NOT PERMITTED >>B1CD
B1AA | SET SYSTEM BUFFER IN OPEN PARM LIST (BC88)
B1B2 | LEVEL = $4F (BF94)
B1B7 | MLI: OPEN <BE78>
B1BA | ERROR? >>B1CB
B1BF | SAVE REFNUM IN READ/WRITE PARMLIST (BE6D)
B1C2 | AND CLOSE PARMLIST (BE6D)
B1C5 | AND GET/SET EOF/MARK LIST (BE7C)
B1C8 | AND EXIT
B1C9 | "FILE TYPE MISMATCH"
B1CC | RETURN
B1CD | "FILE LOCKED"
B1D0 | RETURN

**B1D1 ******* READ NEXT DIRECTORY ENTRY ***********************
B1D1 | FORCE MARK TO START OF THIS BLOCK (BE99)
B1D9 | CHECK ENTRY NUMBER (BC88)
B1DE | LAST ENTRY IN THIS BLOCK? (BC88)
B1E1 | NO >>B1ED
B1E4 | YES, ENTRY @ NEXT TIME (BC88)
B1E7 | BUMP MARK TO NEXT BLOCK (BE99)
B1ED | ---
B1EF | MARK POSITIONED TO PROPER ENTRY YET? >>B1F8
B1F1 | NO, BUMP POINTER TO NEXT ENTRY (BC97)
B1F4 | AND CONTINUE IF STILL FIRST PAGE >>B1ED
B1F6 | JUST ENTERED SECOND PAGE >>B1EA
B1F8 | ADD 4 TO PTR TO ADJUST FOR BLOCK PREFIX
B1FF | MLI: SET MARK <BE78>
B202 | ERROR? >>B21D
B206 | MLI: READ <BE70>
B209 | ERROR? >>B21D
B20B | BUMP ENTRY COUNTER (BC88)
B211 | IS THIS ENTRY VALID?
B213 | NO, SKIP OVER IT >>B1D1
B215 | DECREMENT FILE COUNT (BC99)
B21D | AND RETURN TO CALLER

**B21E ******* EXTERNAL COMMAND HANDLER ****************************
B21E | INDIRECT JMP TO XTRNAE VECTOR >>BE50

**B221 ******* "EXEC" COMMAND ****************************
B221 | IS THIS FILE OPEN ALREADY? <B41F>
B224 | NO >>B250
B226 | YES, EXEC CLOSING? (BE4E)
B229 | NO >>B24C
B22B | SAVE REFNUM (BE7C)
B230 | RESET MARK TO ZERO (BE6C)
B23B | MLI: SET MARK <BE70>
B23E | ERROR? >>B245
B240 | GET REFNUM AGAIN (BE7C)
B243 | GO RESTART THIS EXEC FILE FROM ITS START >>B2C3

******** CLOSE EXEC FILE **********
B245 | PRESERVE CALLER'S AREG
B246 | AND CLOSE THE FILE <B2FB>
B248 | THEN RETURN WITH ERROR
B24C | "FILE BUSY" ERROR
B24F | RETURN

******** CONTINUE EXEC SETUP **********
B258 | EXEC ACTIVE? (BE43)
B25D | NO >>B25A
B25F | YES, CLOSE IP <B2FB>
B263 | ERROR? >>B263
B25A | GET FILE TYPE (BE88)
B25D | SHOULD BE TXT
B25F | IT IS >>B265
B261 | ELSE, "FILE TYPE MISMATCH"
B263 | RETURN WITH ERROR
B26A | RETURN
B265 | MOVE STRINGS TO MAKE ROOM FOR A BUFFER <A1F5>
B268 | NO ROOM? >>B263
B26C | STORE NEW BUFFER ADDRESS IN PARM LIST (BE6C)
B275 | GET COUNT OF OPEN FILES (BEAD)
B278 | NO OTHERS CURRENTLY OPEN? >>B29E

---
BASIC Interpreter (B1) -- V1.1.1 -- 18 JUN 84  NEXT OBJECT ADDR: B27B

******* MAKE EXEC TOPMOST BUFFER ********

B27A OTHERS ARE OPEN...
B27C OPENCOUNT*4 (4 PAGES PER BUFFER)
B27E ADD THIS TO MY BUFFER TO FIND TOP BUFFER (BC88)
B282 SEARCH OPEN FILES TO FIND THE FILE WHICH (BC93)
B285 IS USING THIS BUFFER... >>B28B
B28A IF IT IS NOT FOUND, BREAK!
B28B ---
B28C MOVE THAT FILE TO THE NEW BUFFER INSTEAD (BC93)
B28F GET THAT FILE'S REPNUM ALSO (BC9B)
B297 MALI SET BUFF <BE70>
B29A NO ERRORS? >>B29D
B29C IF ERROR, BREAK!
B29D ---

******* OPEN NEW EXEC FILE ************

B29E SET NEW BUFFER ALLOCATION PAGE (BC88)
B2A1 SET UP OPEN LIST FOR EXEC TOO (BECF)
B2A6 LEVEL = 0 (BP94)
B2AB MALI OPEN (EXEC FILE) <BE70>
B2B0 ---
B2B1 IF ERROR, FREE BUFFER FIRST <A24C>
B2B6 THEN EXIT WITH ERROR
B2B7 SAVE BUFFER FOR EXEC (BECF)
B2BD AND REPNUM TOO (BED6)

******* COMPLETE EXEC COMMAND **********

B2C3 SAVE REPNUM (BED6)
B2C6 AND GET/SET REPNUM (BEC7)
B2CF AND NEWLINE REPNUM (BED2)
B2CC SET "L" VALUE FROM AUXID (BE5F)
B2DD SAVE PATHNAME/AUXID IN OPEN FILE TABLE <B3EB>
B2DD IGNORE MSB FOR END OF LINE CHAR (BED3)
B2DE MALI SET NEWLINE <BE70>
B2EF WAS "P" OR "R" GIVEN ON COMMAND LINE?
B2FA NO >>B2F4
B2FC YES, POSITION TO SPECIFIED STARTING PT <B522>
B2FE NO ERRORS? >>B2F4
B2FF IF ERROR, GO CLOSE EXEC >>B245
B2B4 MARK EXEC ACTIVE
B2FA AND RETURN TO CALLER

BASIC Interpreter (B1) -- V1.1.1 -- 18 JUN 84  NEXT OBJECT ADDR: B2FA

******* CLOSE EXEC FILE *****************

B2FB EXEC ACTIVE? (BE43)
B2FE NO, SKIP IT >>B30B
B30B INDICATE EXEC FILE CLOSING (BE4E)
B305 PICK UP REPNUM FOR EXEC (BC98)
B309 AND GO CLOSE IT <B4A5>
B30B RETURN

B30C "VERIFY" COMMAND ******************

B31C FILE NOT FOUND? >>B347
B311 FILE FOUND, WAS A PATHNAME1 GIVEN?
B313 YES >>B31D
B315 NO,
B317 PRINT "(C) APPLE COMPUTER..." <9F8C>
B31A AND A NEW LINE <9FAB>
B31D THEN EXIT
B31E RETURN

B31F FLUSH ALL OPEN FILES ************

B31F REPNUM = 0 (ALL FILES)
B321 JUMP INTO FLUSH >>B32F

B323 "FLUSH" COMMAND **********

B323 ---
B326 WAS PATHNAME GIVEN?
B328 NO, FLUSH ALL FILES >>B32F
B32A ELSE, LOOK UP NAME IN OPEN FILE LISTS <B41F>
B32D NOT AN OPEN FILE >>B337
B32F SAVE REPNUM IN PARM LIST (BEDE)
B334 MALI FLUSH <BE70>
B337 EXIT

B33B "OPEN" COMMAND **********

B338 ---
B339 LOOK UP NAME IN OPEN FILE LIST <B41F>
B33C NOT CURRENTLY OPEN? >>B34B
B33E ---
B33F IT IS OPEN, "FILE BUSY" ERROR
B342 RETURN
<table>
<thead>
<tr>
<th>BASIC Interpreter (BI) -- V1.1.1 -- 18 JUN 84</th>
<th>NEXT OBJECT ADDR: B342</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADDR</td>
<td>DESCRIPTION/CONTENTS</td>
</tr>
<tr>
<td>B343</td>
<td>&quot;FILE TYPE MISMATCH&quot; ERROR</td>
</tr>
<tr>
<td>B346</td>
<td>RETURN</td>
</tr>
<tr>
<td>B347</td>
<td>&quot;PATH NOT FOUND&quot; ERROR</td>
</tr>
<tr>
<td>B349</td>
<td>---</td>
</tr>
<tr>
<td>B34A</td>
<td>RETURN</td>
</tr>
<tr>
<td>B34B</td>
<td>---</td>
</tr>
<tr>
<td>B34C</td>
<td>ASSUME &quot;L&quot; IS ZERO</td>
</tr>
<tr>
<td>B353</td>
<td>WAS &quot;L&quot; KEYWORD GIVEN?</td>
</tr>
<tr>
<td>B355</td>
<td>YES, USE HIS VALUE &gt;&gt;B35D</td>
</tr>
<tr>
<td>B357</td>
<td>NO, SET &quot;L&quot; TO ZERO (BE68)</td>
</tr>
<tr>
<td>B360</td>
<td>WAS &quot;T&quot; GIVEN?</td>
</tr>
<tr>
<td>B364</td>
<td>YES, USE HIS TYPE &gt;&gt;B36B</td>
</tr>
<tr>
<td>B366</td>
<td>ELSE, DEFAULT TO &quot;TXT&quot;</td>
</tr>
<tr>
<td>B36B</td>
<td>DOES THE FILE ALREADY EXIST? &gt;&gt;B38E</td>
</tr>
<tr>
<td>B36D</td>
<td>NO, &quot;T&quot; GIVEN? IF SO, ERROR &gt;&gt;B347</td>
</tr>
<tr>
<td>B36F</td>
<td>FORCE TYPE = &quot;TXT&quot; (BEB8)</td>
</tr>
<tr>
<td>B374</td>
<td>FULL ACCESS (BEB7)</td>
</tr>
<tr>
<td>B37A</td>
<td>COPY &quot;L&quot; KEYWORD VALUE (BE5F)</td>
</tr>
<tr>
<td>B37D</td>
<td>TO CREATE (BEB6)</td>
</tr>
<tr>
<td>B380</td>
<td>AND SET FILE INFO LISTS (BEB8)</td>
</tr>
<tr>
<td>B389</td>
<td>GO CREATE THE FILE &lt;AD4E&gt;</td>
</tr>
<tr>
<td>B38C</td>
<td>ERROR? &gt;&gt;B349</td>
</tr>
<tr>
<td>B38E</td>
<td>CHECK FILE TYPE (BEB8)</td>
</tr>
<tr>
<td>B391</td>
<td>AGAINST HIS &quot;T&quot; VALUE (BE6A)</td>
</tr>
<tr>
<td>B394</td>
<td>MISMATCH? &gt;&gt;B343</td>
</tr>
<tr>
<td>B396</td>
<td>NO, TYPE = TXT?</td>
</tr>
<tr>
<td>B398</td>
<td>NO &gt;&gt;B3AD</td>
</tr>
<tr>
<td>B39A</td>
<td>YES, GET RECORD LENGTH FROM AUXID (BEB8)</td>
</tr>
<tr>
<td>B393</td>
<td>WAS &quot;L&quot; KEYWORD VALUE GIVEN?</td>
</tr>
<tr>
<td>B395</td>
<td>YES, USE THAT INSTEAD &gt;&gt;B3AD</td>
</tr>
<tr>
<td>B397</td>
<td>OTHERWISE, SAVE AUXID RECORD LEN (BE69)</td>
</tr>
<tr>
<td>B3AD</td>
<td>ALLOCATE A NEW FILE BUFFER &lt;&lt;AIF5&gt;</td>
</tr>
<tr>
<td>B3B8</td>
<td>ERROR? &gt;&gt;B349</td>
</tr>
<tr>
<td>B3B2</td>
<td>GET BUFFER PAGE NO. (BC88)</td>
</tr>
<tr>
<td>B3B5</td>
<td>AND STORE IN OPEN LIST (BECF)</td>
</tr>
<tr>
<td>B3BA</td>
<td>LEVEL = 7 (BF94)</td>
</tr>
<tr>
<td>B3BF</td>
<td>ML: OPEN &lt;BE70&gt;</td>
</tr>
<tr>
<td>B3C2</td>
<td>NO ERRORS? &gt;&gt;B3CB</td>
</tr>
<tr>
<td>B3C4</td>
<td>---</td>
</tr>
<tr>
<td>B3C5</td>
<td>ERROR, FREE BUFFER FIRST &lt;&lt;A24C&gt;</td>
</tr>
<tr>
<td>B3CA</td>
<td>THEN EXIT WITH ERROR CODE</td>
</tr>
<tr>
<td>B3CB</td>
<td>CHECK FILE TYPE AGAIN (BEB8)</td>
</tr>
<tr>
<td>B3CE</td>
<td>&quot;DIR&quot; FILE?</td>
</tr>
<tr>
<td>B3D0</td>
<td>YES &gt;&gt;B3D3</td>
</tr>
<tr>
<td>B3D2</td>
<td>NO</td>
</tr>
<tr>
<td>B3D3</td>
<td>---</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>BASIC Interpreter (BI) -- V1.1.1 -- 18 JUN 84</th>
<th>NEXT OBJECT ADDR: B3D6</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADDR</td>
<td>DESCRIPTION/CONTENTS</td>
</tr>
<tr>
<td>B3D6</td>
<td>SET DIR FLAG ACCORDINGLY (BE47)</td>
</tr>
<tr>
<td>B3D9</td>
<td>USING OPEN COUNT AS AN INDEX (BE4D)</td>
</tr>
<tr>
<td>B3DF</td>
<td>STORE BUFFER LOCATION IN OPEN FILE LIST (BC94)</td>
</tr>
<tr>
<td>B3E5</td>
<td>ALSO, THE RECFNUM (BC9C)</td>
</tr>
<tr>
<td>B3E8</td>
<td>AND BUMP OPEN FILE COUNT AND FALL THRU (BE4D)</td>
</tr>
<tr>
<td>B3EB</td>
<td>************** SAVE FILE NAME/RECLEN IN TABLE ***************</td>
</tr>
<tr>
<td>B3EB</td>
<td>MAKE INDEX FROM RECFNUM*32 BYTES</td>
</tr>
<tr>
<td>B3F1</td>
<td>GET NAME LENGTH (0280)</td>
</tr>
<tr>
<td>B3F4</td>
<td>OR IN DIR FLAG (BE47)</td>
</tr>
<tr>
<td>B3F7</td>
<td>AND STORE IN OPEN FILE NAME LIST (BCFE)</td>
</tr>
<tr>
<td>B3FD</td>
<td>NAME &gt; OR = TO 32 BYTES?</td>
</tr>
<tr>
<td>B3FF</td>
<td>NO... &gt;&gt;B403</td>
</tr>
<tr>
<td>B401</td>
<td>YES, USE 29</td>
</tr>
<tr>
<td>B403</td>
<td>STORE THAT AS A LOOP COUNTER</td>
</tr>
<tr>
<td>B408</td>
<td>COPY &quot;L&quot; KEYWORD VALUE TO NAME LIST TOO (BCFF)</td>
</tr>
<tr>
<td>B411</td>
<td>---</td>
</tr>
<tr>
<td>B412</td>
<td>COPY FILE NAME TO NAME LIST (B28B)</td>
</tr>
<tr>
<td>B418</td>
<td>COPY ALL OF NAME, THEN FALL THRU TO EXIT &gt;&gt;B411</td>
</tr>
<tr>
<td>B41D</td>
<td>************** &quot;NON&quot; AND &quot;NOMON&quot; COMMANDS ***************</td>
</tr>
<tr>
<td>B41D</td>
<td>IGNORE THESE COMMANDS AND</td>
</tr>
<tr>
<td>B41E</td>
<td>RETURN TO CALLER</td>
</tr>
<tr>
<td>B41F</td>
<td>************** LOOKUP OPEN FILENAME **************</td>
</tr>
<tr>
<td>B41F</td>
<td>(RETURNS RECFNUM OF OPEN FILE)</td>
</tr>
<tr>
<td>B41F</td>
<td>---</td>
</tr>
<tr>
<td>B422</td>
<td>WAS PATHNAME GIVEN?</td>
</tr>
<tr>
<td>B424</td>
<td>YES &gt;&gt;B42A</td>
</tr>
<tr>
<td>B426</td>
<td>NO, &quot;SYNTAX ERROR&quot;</td>
</tr>
<tr>
<td>B429</td>
<td>EXIT WITH ERROR</td>
</tr>
<tr>
<td>B42A</td>
<td>ANY FILES CURRENTLY OPEN? (BE4D)</td>
</tr>
<tr>
<td>B42D</td>
<td>NO, CAN'T FIND IT THEN &gt;&gt;B42F</td>
</tr>
<tr>
<td>B42F</td>
<td>YES, CLEAR EXEC FILE CLOSING FLAG (BE4E)</td>
</tr>
<tr>
<td>B432</td>
<td>STORE FILE COUNT AS LOOP COUNTER</td>
</tr>
<tr>
<td>B434</td>
<td>GET NEXT RECFNUM (BC9B)</td>
</tr>
<tr>
<td>B437</td>
<td>COMPARE FILENAMES &lt;&lt;B462&gt;</td>
</tr>
<tr>
<td>B43A</td>
<td>NOT THE ONE? &gt;&gt;B443</td>
</tr>
<tr>
<td>B43C</td>
<td>ELSE, WE'VE GOT IT</td>
</tr>
<tr>
<td>B43E</td>
<td>PICK UP APPROPRIATE RECFNUM (BC9B)</td>
</tr>
<tr>
<td>B441</td>
<td>---</td>
</tr>
<tr>
<td>B442</td>
<td>AND RETURN WITH IT</td>
</tr>
<tr>
<td>B443</td>
<td>ELSE, NOT IT, TRY NEXT ONE</td>
</tr>
<tr>
<td>B446</td>
<td>AND CONTINUE LOOPING &gt;&gt;B432</td>
</tr>
</tbody>
</table>
BASIC Interpreter (BI) -- V1.1.1 -- 18 JUN 84
NEXT OBJECT ADDR: B446

ADDRESS DESCRIPTION/CONTENT

B448  CAN'T FIND IT, IS EXEC ACTIVE? (BE43)
B44B  NO, THEN WE MUST GIVE UP >>B45E
B450  IS HE LOOKING FOR EXEC FILE? <B462>
B453  NO, GIVE UP >>B45E
B457  YES, EXEC FILE CLOSING (BE4E)
B45C  AND RETURN WITH EXEC'S REPNUM >>B43E
B45E  "FILE NOT OPEN" ERROR
B461  RETURN WITH ERROR CODE

B462  ********** COMPARE FILENAMES ********************

B468  REPNUM*32 FOR FILENAME INDEX
B46B  PICK UP DIR FLAG FROM THIS ENTRY (BCFE)
B470  SAME LENGTH AS HIS FILENAME? (B2B0)
B473  NO, CAN'T BE IT THEN >>B498
B476  MAKE SURE LENGTH DOES NOT EXCEED 29
B479  IF IT DOES, ONLY LOOK AT FIRST 29
B47C  USE $AA AS LOOP COUNTER
B481  COPY "L" OF THIS FILE TO KEYWORD (BCA4)
B484  ***
B487  COMPARE NAMES (B2B0)
B491  NO MATCH? EXIT WITH Z FLAG CLEAR >>B498
B492  MATCH, EXIT WITH Z FLAG SET

B499  ********** "CLOSE" COMMAND ********************

B49E  ***
B49C  PATHEMAGEL GIVEN?
B49E  NO, CLOSE ALL FILES >>B4F2
B4A0  YES, LOOK IT UP IN OPEN FILE TABLES <<B41P>
B4A3  NOT FOUND? >>B441
B4A5  FOUND IT, STORE REPNUM IN CLOSE LIST (BDEE)
B4AB  MARK BUFFER PAGE FREE (BCB0)
B4AE  EXEC CLOSING? (BE4E)
B4B1  YES...NO NEED TO COMPRESS LISTS >>B4CF
B4B3  GET OPEN COUNT (LAST OPENED FILE NO.) (BE4D)
B4B7  SWAP BUFFERS (BC93)
B4C0  AND REPNUMS WITH THE LAST OPENED FILE (BC9B)
B4C2  ***
B4D1  LEVEL = 0 (BF94)
B4D6  MLI: CLOSE <BE70>
B4D9  ERROR? >>B502
B4DB  RELEASE THE BUFFER <A24C>
B4DE  EXEC FILE CLOSING? (BE4E)
B4E1  NO >>B48E
B4E5  YES, EXEC NO LONGER ACTIVE (BE43)
B4EC  AND NO LONGER CLOSING (BE4E)
B4ED  RETURN TO CALLER

BASIC Interpreter (BI) -- V1.1.1 -- 18 JUN 84
NEXT OBJECT ADDR: B46E

ADDRESS DESCRIPTION/CONTENT

B44E  DROP OPEN FILE COUNT (BE4D)
B44F  AND EXIT

B4F2  ********** CLOSE ALL OPEN FILES ********************

B4F5  ANY FILES OPEN? (BE4D)
B4F7  NO >>B503
B4FD  YES, EXEC NOT CLOSING (BE4E)
B500  IF THAT WORKS, START ALL OVER AGAIN >>B4F2
B502  EXIT WHEN ALL ARE CLOSED

B503  ---
B505  SET CLOSE REPNUM TO ZERO (ALL FILES) (BEDE)
B50A  LEVEL = 7 (LEVEL 8 FILES ALREADY CLOSED) (BF94)
B50F  EXIT THRU MLI: CLOSE >>BE70

B512  ********** "POSITION" COMMAND ********************

B512  LOOKUP NAME OF FILE <B41F>
B515  NOT OPEN? >>B57F
B517  SET REPNUM IN READ/WRITE PARMLIST (BED6)
B51A  AND SET NEWLINE LIST (BED2)
B51D  DIR FILE? (BE47)
B520  YES, GET OUT RIGHT NOW1 >>B580
B522  "P" OR "R" GIVEN? (BE57)
B527  NO, INVALID PARM >>B57D
B529  BOTH GIVEN?
B52B  YES, INVALID PARM >>B57D
B52D  JUST "R" GIVEN?
B52F  NO, JUST "P" >>B35D
B531  JUST "R", COPY "R" VALUE TO "P" (BE65)
B534  ("P" AND "R" ARE ALIASED) (B653)
B53D  SET COUNT TO 239. (MAXIMUM LINE LEN)
B54C  BUFFER IS AT $200 (BED0)
B54F  ***
B551  NEW LINE CHAR IS EITHER $0D OR $0A (BED3)
B556  MLI: SET NEWLINE <BE70>
B559  ERROR? >>B57F

***** SKIP LINES BY READING THEM *****

B55B  ---
B55E  "P" = $07 (BE64)
B562  YES, DONE >>B580
B564  ELSE...
B566  MLI: READ NEXT FIELD (LINE) <BE70>
B569  ERROR? >>B57F
B56E  DECREMENT "P" VALUE BY ONE
BASIC Interpreter (BI) -- V1.1.1 -- 18 JUN 84  NEXT OBJECT ADDR: B57B

B57B AND GO CHECK IT AGAIN >>B55B
B57D "INVALID PARAMETER" ERROR
B57F --
B580 EXIT TO CALLER

B581 ********** COMPUTE NEW FILE POSITION **********************
(COMPUTES ABSOLUTE FILE POSITION MARK)
B581 ACCUM = CURRENT RECORD LENGTH (BCA4)
B595 MARK = 0 (BEC9)

********** MARK = "R" * RECLEN **********
B59E SHIFT "R" VALUE RIGHT (B666)
B5A6 IF LOW BIT OFF, NO ADD >>B5BF
B5A9 ADD ONE INSTANCE OF RECLEN TO MARK (BCAF)
B5B8 OVERFLOW? >>B5D2
B5BD ACCUM OVERFLOW? >>B5D2
B5BF SCALE ACCUM (MULTIPLIER) UP BY 2 (BCAF)
B5C9 IF "R" NOW ZERO... (B665)
B5CE CONTINUE LOOPING >>B59E
B5D1 ELSE, EXIT TO CALLER
B5D2 "RANGE ERROR"
B5D5 RETURN

B5D6 ********** "READ" COMMAND **************
B5D6 LOOK UP FILE NAME <B41F>
B5D9 NOT OPEN? >>B62B
B5DB ITS OPEN, STORE RENUM IN READ/WRITE... (BED6)
B5DB GET/SET... (BEC7)
B5EB AND SET NEWLINE PARMLISTS (BED2)
B5ED DIR FILE? (BE47)
B5E7 YES, SPECIAL HANDLING REQUIRED >>B62C
B5E9 NO, PRE-POSITION FOR "B", "T", OR "R" <B666>
B5EC ERROR POSITIONING? >>B62B
B5EE ASSUME "L" = 239.
B5F5 "L" GIVEN?
B5F7 NO >>B66C
B5F9 YES, USE HIS "L" VALUE (B65F)
B5FF UNLESS ITS >256 >>B661
B603 OR >239. >>B661
B607 DOUBLE QUIT IT SO COMMAS COME THRU (0200)
B60A READ INTO $201
B60C IF NO "L", READ TO $200 (BED7)
B612 NL CHAR = $8D/$8D (OR NONE IF "L") (BED3)
B621 ML: SET NEWLINE <BE70>
B624 ERROR? >>B62B
B626 --

B62B MARK INPUT "READ" FILE ACTIVE (BE44)
B62B AND RETURN

********** READ DIR FILE **************
B62C SET READ/WRITE LIST RENUM (BED6)
B62F AND GET/SET LIST RENUM (BEC7)
B634 READING TO $259 (BED7)
B63E INIT CAT FLAG TO FIRST LINE VALUE (BE4F)
B644 "R" GIVEN?
B647 NO, DONE >>B626
B64B YES, ZERO OUT MARK (BEC8)
B656 ML: REWIND FILE <BE70>
B659 ERROR? >>B660
B65D MARK INPUT FILE ACTIVE (BE44)
B660 AND EXIT

B661 ********** "RANGE ERROR" **********************
B665 EXIT TO CALLER

B666 ********** PRE-POSITION FOR I/O *********************
B666 --
B669 "B", "T", OR "R" GIVEN?
B66B NO, EXIT >>B6AF
B66D "R"?
B66F NO >>B67B
B671 YES, COMPUTE ABSOLUTE POSITION <B581>
B674 ERROR? >>B661
B676 NO, SET MARK TO NEW POSITION <B6A8>
B679 ERROR? >>B6B8
B67B "R" GIVEN? (B657)
B680 NO >>B677
B682 SKIP LINES UNTIL "F" = 0 <B53D>
B685 ERROR? >>B6B8
B687 "B" GIVEN? (B657)
B68C NO >>B6AF
B690 ML: GET MARK <BE70>
B693 ERROR? >>B6B8
B699 ADD "$" VALUE TO CURRENT MARK (B65A)
B69C (3 BYTE ADD) (BEC8)
B6A6 OVERFLOW? >>B661
B6AB --
B6A9 ML: SET MARK <BE70>
B6AD ERROR? >>B6B0
B6AF --
B6B0 --
B6B2 EXIT TO CALLER
BASIC Interpreter (B1) -- V1.1.1 -- 18 JUN 84  NEXT OBJECT ADDR: B6B2

ADDR DESCRIPTION/CONTENTS

B6B3 *********** "WRITE" COMMAND ****************************

B6B3 LOOKUP OPEN FILE NAME <B41F>
B6B6 NOT AN OPEN FILE? >>B6C8
B6B8 STORE READ/WRITE REFNUM (BED6)
B6BB AND GET/SET REFNUM (BEC7)
B6BE AND NEWLINE REFNUM IN PARM LISTS (BED2)
B6C1 DIR FILE? (BE47)
B6C4 NO, OK >>B6CA

B6C6 YES, "FILE LOCKED" ERROR
B6C8 ---
B6C9 EXIT TO CALLER

B6CA DATA BUFFER AT $200
B6D4 PRE-POSITION FOR "B", "F", AND "R" <B666>
B6D7 NO ERRORS? >>B6ED
B6D9 WAS ERROR A RANGE ERROR?
B6DB NO, REAL ERROR >>B6C8
B6DD YES, MY RANGE ERROR OR MLI'S?
B6DF MINE... >>B6C8
B6E1 MLI'S...SET EOF FARTHER INTO FILE
B6E3 MLI: SET EOF <BE78>
B6E6 ERROR? >>B6C8
B6E8 AND THEN TRY AGAIN TO SET MARK <B676>
B6EB ERROR? THEN I GIVE UP >>B6CB
B6ED BUFFER IS AT HIMEM
B6EF INDICATE OUTPUT "WRITE" FILE ACTIVE (BE45)
B6FD RETURN TO CALLER

B6FE *********** "APPEND" COMMAND ****************************

B6FF ---
B6F0 LOOK UP NAME IN OPEN FILE LIST <B41F>
B702 FOUND IT? >>B718
B705 NO, OPEN IT FIRST <B33B>
B708 ERROR? >>B71E
B70A NO, REFNUM NON-ZERO? (BED0)
B70D YES, OK >>B711
B70F ELSE, BREAK111
B710 ---
B718 REFNUM TO READ/WRITE PARM LIST (BED6)
B71A AND GET/SET LIST (BEC7)
B717 DIR FILE? (BE47)
B71A NO >>B720

B71C YES, "FILE LOCKED"
B71E ---
B71F EXIT TO CALLER

B720 PICK UP "L" VALUE (BE5F)
B729 DID USER SPECIFY ONE?
B72B YES... >>B733
B72D NO, USE FILE'S CURRENT "L" VALUE (BE89)
B733 ---
B738 COMPUTE REFNUM*32 FOR INDEX INTO
B739 FILE NAME TABLE
B73E SAVE CURRENT "L" VALUE IN OPEN FILE (BCF0)
B741 NAME TABLE AND IN CURRENT RECLLEN (BCC4)
B744 MLI: GET EOF <BE78>
B750 ERROR? >>B71E
B752 IS "L" VALUE < 2? (NO SPECIFIC "L") (BCA5)
B755 NO >>B75E
B75C YES >>B763
B75E NO, FORCE TO RECORD BOUNDARY <B676>
B761 ERROR? >>B71E
B763 ELSE, GO SET EOF=MARK/OUTPUT FILE ACTIVE >>B6E1

B766 *********** FORCE TO EVEN RECORD BOUNDARY ****************************
(FIND RECORD NUMBER OF THIS POSITION)

B766 ---
B768 COPY EOF TO ACCUM (BCC7)
B771 CLEAR MSB'S (BCC2)
B777 GET READY FOR A 24 BIT DIVIDE
B779 DIVIDE EOF BY... <AAD7>
B786 RECORD LENGTH (BCC4)
B79B ---
B7A1 WAS THERE A REMAINDER? (BCB3)
B7A5 NO, OK... >>B7CF
B7AB YES, CURRENT RECORD LEN LESS REMAINDER (BCC2)
B7B8 PLUS OLD EOF MARK (BCC8)
B7C2 GIVES NEW EOF ON AN EVEN RECORD BOUNDARY (BCC9)
B7CD "RANGE ERROR" POSSIBLE IF OVERFLOW OCCURS
B7CF RETURN TO CALLER

B7DB *********** GET FILE INFO ****************************

B7DB ---
B7D8 SET NUMBER OF PARMS (10)
B7D5 MLI CODE FOR GET FILE INFO
B7D7 GO DO IT >>B72E
BASIC Interpreter (BI) -- V1.1.1 -- 18 JUN 84  NEXT OBJECT ADDR: B7D7

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

B7D9 ********** SET FILE INFO *******************************

B7D9 MODIFIED TIME/DATA = 0
B7E7 SET NUMBER OF PARMS (7)
B7EC MLI CODE FOR SET FILE INFO
B7EE EXIT THRU MLI: GET/SET FILE INFO >>BE70

B7F1 ********** BI I/O INDICTION VECTORS ***************

B7F1 DOSOUT VECTOR >>BE38
B7F4 DOSIN VECTOR >>BE3A

B7F7 ********** STATE I/O VECTORS TABLE ***************

B7F7 IMMEDIATE MODE (STATE=0) CSWL/KSWL
B7FB DEFERRED MODE (STATE=4) CSWL/KSWL
B7FF (STATE=8) CSWL/KSWL
B803 (STATE=C) CSWL

B805 ********** SYSCTLR *******************************
LSB'S OF MLI CALL PARAMETER LISTS IN THE
BI GLOBAL PAGE ($BEXX)

B805 CREATE: $A0 DESTROY: $A0 RENAME: $AF
B808 SPI: $B4 SPI: $B4 ONLINE: $C6
B80B SPFX: $AC SPFX: $AC OPEN: $CB
B80E NEWLINE:$D1 READ: $D5 WRITE: $D5
B811 CLOSE: $D8 FLUSH: $D8 SMARK: $C6
B814 GMARK: $C6 SEOF: $C6 GEOM: $C6
B817 SMBUF: $C6 GBUF: $C6

B819 ********** APPLESOFT TOKENS ****************************

TOKENS REQUIRING SPECIAL ATTENTION HAVE
THEIR MSB OFF AND ARE AN OFFSET FROM A
JMP IN THE TRACE HANDLER IN THE BI

B819 FIRST IS $86 (END)
B823 CALL
B833 TRACE, NOTRACK, NORMAL
B837 INVERSE, FLASH
B83F RESUME
B843 LET, IF
B853 PRINT, LIST

B859 ********** COMMAND NAME TABLES ****************************

OFFSETS TO LAST CHARACTER OF EACH COMMAND
NAME IN THE COMMAND NAME TABLE BELOW.
COMMANDS ARE ARRANGED ACCORDING TO LENGTH
WITH THREE BYTE NAMES FIRST. IF THE MSB
OF AN INDEX IS ON, THEN THIS IS THE LAST

NAME OF THE GIVEN LENGTH (NEXT WILL BE
ONE BYTE LONGER).

B859 01 IN# 02 PR# 03 CAT
B85C 04 PRE 05 BYE 06 RUN
B85F 07 BRUN 08 EXEC 09 LOAD
B862 0A LOCK 0B OPEN 0C READ
B865 0D SAVE 0E BLOAD 0F BSAVE
B868 10 CHAIN 11 CLOSE 12 FLUSH
B86B 13 NOMON 14 STORE 15 WRITE
B86E 16 APPEND 17 CREATE 18 DELETE
B871 19 PREFIX 1A RENAME 1B UNLOCK
B874 IC VERIFY 1D CATALOG 1E RESTORE
B877 1F POSITION

B878 " REDL" "DELETESTORENAMEBRUNLO"
B879 "NEEDCREATEDRFRESHENVIRONMENT"
B87A "CKCHAIN#FLUSHREADPOSITIONONOMONPR#"
B87B "PREFIXCLOSEAPPEND"

B889 ********** COMMAND HANDLER ADDRESS TABLE *************
ADDRESSES OF THE COMMAND HANDLER ROUTINES
FOR EACH COMMAND IN THE ORDER GIVEN ABOVE.

B889 (EXTERNAL)
B88D IN#
B88D PR#
B88F CKT
B891 PRE
B893 BYE
B895 RUN
B897 BRUN
B89F EXEC
B8AF LOAD
B8B0 LOCK
B8B4 OPEN
B8B8 READ
B8B9 SAVE
B8B9 BLOAD
B8BB BSAVE
B8BD CHAIN
B8BF CLOSE
B8C0 FLUSH
B8C0 NOMON
B8C4 STORE
B8CA WRITE
B8CC APPEND
B8CD CREATE
B8CE DELETE
B8CF PREFIX
B8D1 RENAME
B92B ************ PERMITTED KEYWORDS FOR CMDS ***************

TWO BYTES PER COMMAND IN THE ORDER ABOVE.
EACH ENTRY HAS 16 BIT SETTINGS FOR THE
PARAMETERS PERMITTED ON THAT COMMAND.
$0000 = FETCH PREFIX, PATHNAME OPTIONAL
$0000 = SLOT FOR PR# OR IN#
$0000 = DEFERRED COMMAND ONLY
$0000 = FILENAME IS OPTIONAL
$0000 = IF FILE NOT FOUND, CREATE IT
$0000 = "=" (FILE TYPE) PERMITTED
$0000 = PATHNAME2 (RENAME) PERMITTED
$0000 = PATHNAME1 EXPECTED
$0000 = "A" (ADDRESS) PERMITTED
$0000 = "B" (BYTE) PERMITTED
$0000 = "E" (END ADDRESS) PERMITTED
$0000 = "L" (LENGTH) PERMITTED
$0000 = "N" (LINE NO.) PERMITTED
$0000 = "S" AND/OR "D" (SLOT/DRIVE)
$0000 = "P" (FIELD) PERMITTED
$0000 = "R" (RECORD) PERMITTED
("V" IS IGNORED)

C P S D E F N T P P A B E L @ S F R
O X Q O P O W T T T D
M T E F P H H H H
N | R T I | 2 1
D | | | | |

B92B IN# X X X X X X X X
B92D PR# X X X X X X X X
B92F CAT X X X X X X X X
B931 FRE X X X X X X X X
B933 BYE X X X X X X X X
B935 RUN X X X X X X X X
B937 BRUN X X X X X X X X
B939 EXEC X X X X X X X X
B93B LOAD X X X X X X X X
B93D LOCK X X X X X X X X
B93F OPEN X X X X X X X X
B941 READ X X X X X X X X
B943 SAVE X X X X X X X X
B945 BLOAD X X X X X X X X
B947 BSAVE X X X X X X X X

B968 ************ KEYWORD NAME TABLE ***********************

B96B 'ABELSDFRV6'

B975 ************ 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)

B975 ---

B97F ************ KEYWORD SIZE/OFFSET TABLE ***********************

LOW 2 BITS = SIZE-1 OF VALUE IN BYTES
HIGH 6 BITS = OFFSET TO LAST BYTE OF VALUE
FROM $BE58

B97F A1: 2 BYTES AT +1
B980 B: 3 BYTES AT +4
B981 E: 2 BYTES AT +6
B982 L: 2 BYTES AT +8
B983 S: 1 BYTE AT +9
B984 D: 1 BYTE AT +A
B985 F: 2 BYTES AT +C
B986 R: 2 BYTES AT +E
B987 V: 1 BYTE AT +10 (IGNORED)
B998 $: 2 BYTES AT +11

B989 ************ FILE TYPES TABLES ***********************

FILE TYPE CODES, GIVEN IN REVERSE ORDER
TO FILE TYPE NAMES WHICH FOLLOW.
BASIC Interpreter (BI) -- V1.1.1 -- 18 JUN 84  NEXT OBJECT ADDR: B989
ADDR   DESCRIPTION/CONTENTS

B989 $FF = "SYS"
B98A $FE = "REL"
B98B $FD = "VAR"
B98C $FC = "BAS"
B98D $FB = "IVR"
B98E $FA = "INT"
B98F $F8 = "CMD"
B990 $F6 = "DIR"
B991 $06 = "BIN"
B992 $04 = "TXT"
B993 $0F = "PAS"
B994 $1A = "AWP"
B995 $1B = "ASP"
B996 $19 = "ADB"

B997 ---
B997 'ADBA$PAWPPASTXTBSINDIRCMDTINTIVRBSVARRELSYS'

B9C1 ********** MONTH TABLE **********************
B9C1 'JANFEBMARAPRMAYJUNJULAugSEPCTNOVDEC'
B9E5 'NO DATE'

B9EE ********** MLEKTB1  **********************
MLI ERROR CODES WHICH HAVE BI EQUIVALENTS

B9EE ---

BAA1 ********** BIEKTB1  **********************
BI EQUIVALENTS TO MLI ERROR CODES ABOVE
(IF MLI CODE NOT FOUND, MAPS TO LAST CODE
IN THIS TABLE, $08 "I/O ERROR")

BAA1 ---

BA15 ********** INDEXES TO PACKED MESSAGES **********************
BY BI ERROR NUMBER

BA15 ---

BA29 ********** COMMON LETTERS IN MESSAGES **********************
BA29 ---
BA29 'ACDEFLMNORTU'

BASIC Interpreter (BI) -- V1.1.1 -- 18 JUN 84  NEXT OBJECT ADDR: BA38
ADDR   DESCRIPTION/CONTENTS

BA38 ********** LESS COMMON LETTERS **********************
BA38 ---
BA39 'BGKPSVWXYZ().,:'

BA48 ********** PACKED MESSAGES **********************
BA48 "COPYRIGHT APPLE COMPUTER"
BA58 "NAME"
BA5B TAB($10)
BA5D "TYPE BLOCKS"
BA66 TAB($1E)
BA68 "MODIFIED"
BA6C TAB($2F)
BA6E "CREATED"
BA72 TAB($48)
BA74 "ENDFILE SUBTYPE"
BA7E "BLOCKS FREE:"
BA86 TAB($16)
BA88 "BLOCKS USED:"
BA91 TAB($2C)
BA93 "TOTAL BLOCKS:"
BA9C "RANGE ERROR"  ERROR=52
BA93 "NO DEVICE CONNECTED"  ERROR=53
BAAE "WRITE PROTECTED"  ERROR=54
BA7E "END OF DATA"  ERROR=55
BABD "PATH NOT FOUND"  ERROR=6,7
BAC6 "I/O ERROR"  ERROR=58
BACC "DISK FULL"  ERROR=59
BAD2 "FILE LOCKED"  ERROR=5A
BAD9 "INVALID PARAMETER"  ERROR=5B
BAE3 "RAM TOO LARGE"  ERROR=5C
BAF9 "FILE TYPE MISMATCH"  ERROR=5D
### BASIC Interpreter (BI) -- V1.1.1 -- 18 JUN 84 NEXT OBJECT ADDR: BAF8

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAF8</td>
<td>ERROR=6E</td>
</tr>
<tr>
<td>BB07</td>
<td>&quot;NOT DIRECT COMMAND&quot; ERROR=6F</td>
</tr>
<tr>
<td>BB11</td>
<td>&quot;SYNTAX ERROR&quot; ERROR=810</td>
</tr>
<tr>
<td>BB19</td>
<td>&quot;DIRECTORY FULL&quot; ERROR=811</td>
</tr>
<tr>
<td>BB21</td>
<td>&quot;FILE NOT OPEN&quot; ERROR=812</td>
</tr>
<tr>
<td>BB29</td>
<td>&quot;DUPLICATE FILE NAME&quot; ERROR=813</td>
</tr>
<tr>
<td>BB34</td>
<td>&quot;FILE BUSY&quot; ERROR=814</td>
</tr>
<tr>
<td>BB3B</td>
<td>&quot;FILE(S) STILL OPEN&quot; ERROR=815</td>
</tr>
</tbody>
</table>

### Variables

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>BB47</td>
<td>NUMBER OF PAGES TO ALLOCATE/FREE</td>
</tr>
<tr>
<td>BB4B</td>
<td>NOT USED</td>
</tr>
<tr>
<td>BB49</td>
<td>TOP OF BUFFERS FOR GARBAGE COLLECTION</td>
</tr>
<tr>
<td>BB4A</td>
<td>BOTTOM OF BUFFERS</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>BB4B</td>
<td>SBB4B-SC7A NOT USED</td>
</tr>
</tbody>
</table>

### Variables

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>BC7B</td>
<td>SAVED HIMEM VALUE DURING CHAIN LOAD</td>
</tr>
<tr>
<td>BC7C</td>
<td>GC: HI RANGE - WORK AREA SIZE</td>
</tr>
<tr>
<td>BC7D</td>
<td>GC: WORK AREA MSB</td>
</tr>
<tr>
<td>BC7E</td>
<td>GC: NUMBER OF PAGES IN WORK AREA</td>
</tr>
<tr>
<td>BC7F</td>
<td>GC: LO RANGE (START OF STRINGS TO COPY)</td>
</tr>
<tr>
<td>BC80</td>
<td>GC: HI RANGE (END OF STRINGS TO COPY)</td>
</tr>
<tr>
<td>BC81</td>
<td>ARRAYS START LSB</td>
</tr>
<tr>
<td>BC82</td>
<td>ARRAYS ENDING MSB+1</td>
</tr>
<tr>
<td>BC83</td>
<td>GC: START OF STRING AREA (ALSO PGM START)</td>
</tr>
<tr>
<td>BC85</td>
<td>GC: END OF STRING AREA</td>
</tr>
<tr>
<td>BC87</td>
<td>MSB ADJUST FACTOR FOR STRING POINTERS</td>
</tr>
<tr>
<td>BC88</td>
<td>PAGE FOLLOWING BLOCK BUFFER</td>
</tr>
</tbody>
</table>

### Stored Variables File Header

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>BC89</td>
<td>COMBINED LEN OF SIMPLE/ARRAY VARS</td>
</tr>
<tr>
<td>BC90</td>
<td>LEN OF SIMPLE VARS ONLY</td>
</tr>
<tr>
<td>BC8D</td>
<td>HIMEM WHEN VARS WERE COMBINED</td>
</tr>
</tbody>
</table>

### Combined Vars/Strings

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>BC92</td>
<td>LENGTH OF STRINGS ONLY</td>
</tr>
</tbody>
</table>

**Note:**
- **BC94**: OPEN FILES' BUFFER MSBS
- **BC9B**: OPEN EXEC FILE BUFFER MSB
- **BC9C**: OPEN FILES' REFERENCE NUMBERS
- **BCA3**: OPEN EXEC FILE REFNUM
- **BCA4**: CURRENT RECORD LENGTH
- **BCA6**: NOT USED
- **BCA9**: CHARACTER TO FLUSH WHEN PARSING (BLANK)
- **BCAA**: MAXIMUM LENGTH TO PARSE
- **BCAB**: ADDRESS OF COMMAND HANDLING ROUTINE
- **BCAD**: SIZE OF KEYWORD VALUE -1 IN BYTES
- **BCAE**: OFFSET INTO KEYWORD PARM TO LAST BYTE
- **BCAF**: GENERAL PURPOSE 4 BYTE ACCUMULATOR
- **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

### Pathname 1 Buffer

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>BCBC</td>
<td>COMMAND OR PATH LENGTH</td>
</tr>
<tr>
<td>BCBD</td>
<td>TXBUF (COMMAND OR PATHNAME STRING)</td>
</tr>
<tr>
<td>BCFD</td>
<td>NOT USED</td>
</tr>
</tbody>
</table>

### Open File Name Table

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>BCFE</td>
<td>FILE 0: LENGTH OF NAME</td>
</tr>
<tr>
<td>BCFF</td>
<td>FILE 0: L VALUE LSB</td>
</tr>
<tr>
<td>BD00</td>
<td>FILE 0: L VALUE MSB</td>
</tr>
<tr>
<td>BD01</td>
<td>FILE 0: START OF NAME STRING</td>
</tr>
</tbody>
</table>

**Note:** (FILE NAME IS STORED BACKWARDS)

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>BD0E</td>
<td>LAST 2 BYTES NOT USED</td>
</tr>
</tbody>
</table>
**BASIC INTERPRETER GLOBAL PAGE**

This page of memory is rigidly defined by the ProDOS BI. Fields given here will not move in later versions of ProDOS and may be referenced by external, user-written programs. Future additions to the global page may be made in areas which are marked "Not used".

<table>
<thead>
<tr>
<th>ADDR</th>
<th>LABEL</th>
<th>CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>BE00-BE02</td>
<td>BI.ENTRY</td>
<td>JMP to WARMDOSS (BI warmstart vector).</td>
</tr>
<tr>
<td>BE03-BE05</td>
<td>DOSCMD</td>
<td>JMP to SYNTAX (BI command line parse and execute).</td>
</tr>
<tr>
<td>BE06-BE08</td>
<td>EXTRNCMD</td>
<td>JMP to user-installed external command parser.</td>
</tr>
<tr>
<td>BE09-BE0B</td>
<td>ERROUT</td>
<td>JMP to BI error handler.</td>
</tr>
<tr>
<td>BE0C-BE0E</td>
<td>PRINTERR</td>
<td>JMP to BI error message print routine.</td>
</tr>
<tr>
<td>BE10-BE10</td>
<td>ERRCODE</td>
<td>Place error number in A-register.</td>
</tr>
<tr>
<td>BE11-BE1F</td>
<td>OUTVEC</td>
<td>ProDOS error code (also at $5E, Applesoft ONERR code).</td>
</tr>
<tr>
<td>BE1A-BE22</td>
<td>INVEC</td>
<td>Default output vector in monitor and for each slot (1-7).</td>
</tr>
<tr>
<td>BE38-BE3A</td>
<td>VECTOUT</td>
<td>Default input vector in monitor for each slot (1-7).</td>
</tr>
<tr>
<td>BE3A-BE3C</td>
<td>VECTIN</td>
<td>Current output vector.</td>
</tr>
<tr>
<td>BE34-BE36</td>
<td>VDOSIC</td>
<td>Current input vector.</td>
</tr>
<tr>
<td>BE36-BE38</td>
<td>VDOSIO</td>
<td>BI's output intercept address.</td>
</tr>
<tr>
<td>BE39-BE3B</td>
<td>VSYSIO</td>
<td>BI's input intercept address.</td>
</tr>
<tr>
<td>BE3C-BE3E</td>
<td>DEFGLT</td>
<td>BI's internal redirection by STATE.</td>
</tr>
<tr>
<td>BE3F-BE41</td>
<td>DEPDRV</td>
<td>Default slot.</td>
</tr>
<tr>
<td>BE3E-BE40</td>
<td>DEPDRV</td>
<td>Default drive.</td>
</tr>
<tr>
<td>BE40-BE42</td>
<td>PREGA</td>
<td>A-register savearea.</td>
</tr>
<tr>
<td>BE41-BE43</td>
<td>PREGX</td>
<td>X-register savearea.</td>
</tr>
<tr>
<td>BE42-BE44</td>
<td>PREGY</td>
<td>Y-register savearea.</td>
</tr>
<tr>
<td>BE43-BE45</td>
<td>TRACE</td>
<td>Applesoft TRACE is enabled flag (MSB on).</td>
</tr>
<tr>
<td>BE44-BE46</td>
<td>STATE</td>
<td>Current intercept state. 0 = immediate command mode. 0 = deferred.</td>
</tr>
<tr>
<td>BE45-BE47</td>
<td>EXACTV</td>
<td>EXEC file active flag (MSB on).</td>
</tr>
<tr>
<td>BE46-BE48</td>
<td>IFILACTV</td>
<td>READ file active flag (MSB on).</td>
</tr>
<tr>
<td>BE47-BE49</td>
<td>OFILACTV</td>
<td>WRITE file active flag (MSB on).</td>
</tr>
<tr>
<td>BE48-004E</td>
<td>PFILACTV</td>
<td>PREFIX read active flag (MSB on).</td>
</tr>
<tr>
<td>BE49-004F</td>
<td>DIRFLG</td>
<td>File being READ is a DIR file (MSB on).</td>
</tr>
<tr>
<td>BE4F</td>
<td>EDIRFLG</td>
<td>End of directory flag (no longer used).</td>
</tr>
<tr>
<td>BE50-0051</td>
<td>STRINGS</td>
<td>String space count used to determine when to garbage collect.</td>
</tr>
<tr>
<td>BE51</td>
<td>TRUFSR</td>
<td>Buffered WRITE data length.</td>
</tr>
<tr>
<td>BE52</td>
<td>INPTR</td>
<td>Command line assembly length.</td>
</tr>
<tr>
<td>BE53</td>
<td>CHRLAST</td>
<td>Previous output character (for recursion check).</td>
</tr>
<tr>
<td>BE54</td>
<td>OPENCNT</td>
<td>Number of files open (not counting EXEC).</td>
</tr>
<tr>
<td>BE55</td>
<td>YXFILE</td>
<td>EXEC file being closed flag (MSB on).</td>
</tr>
<tr>
<td>BE56</td>
<td>CATFLAG</td>
<td>Line type to format next in DIR file READ.</td>
</tr>
<tr>
<td>BE57-BE59</td>
<td>XTRNADDR</td>
<td>External command handler address.</td>
</tr>
<tr>
<td>BE5A</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>XCNUM</td>
<td>Number of command:</td>
</tr>
<tr>
<td>$00</td>
<td>$0A</td>
<td>OPEN</td>
</tr>
<tr>
<td>$01</td>
<td>$0B</td>
<td>READ</td>
</tr>
<tr>
<td>$02</td>
<td>$0C</td>
<td>SAVE</td>
</tr>
<tr>
<td>$03</td>
<td>$0D</td>
<td>BLOAD</td>
</tr>
<tr>
<td>$04</td>
<td>$0E</td>
<td>BSAVE</td>
</tr>
<tr>
<td>$05</td>
<td>$0F</td>
<td>CHAIN</td>
</tr>
<tr>
<td>$06</td>
<td>$10</td>
<td>CLOSE</td>
</tr>
<tr>
<td>$07</td>
<td>$11</td>
<td>FLUSH</td>
</tr>
<tr>
<td>$08</td>
<td>$12</td>
<td>NOMON</td>
</tr>
<tr>
<td>$09</td>
<td>$13</td>
<td>STORE</td>
</tr>
<tr>
<td></td>
<td>$1E</td>
<td>POSITION</td>
</tr>
</tbody>
</table>

**BE54-BE55 PBITS**  
Permitted command operands bits:

- **$0000** Prefix needed. Pathname optional.
- **$0100** Slot number only (PR# or IN#).
- **$0200** Deferred command.
- **$0300** Deferred file name optional.
- **$0400** If file does not exist, create it.
- **$0500** T: file type required.
- **$0600** Second file name required.
- **$0700** First file name required.
- **$0800** AD: address keyword permitted.
- **$0900** B: byte offset permitted.
- **$0A00** E: ending address permitted.
- **$0B00** L: length permitted.
- **$0C00** Q: line number permitted.
- **$0D00** S or D: slot/drive permitted.
- **$0E00** F: field permitted.
- **$0F00** R: record permitted.

(V always permitted but ignored.)

**BE56-BE57 PBITS**  
Operands found on command line. Same bit assignments as above.

- **BE58-BE59** VADDR: A keyword value.
- **BE5A-BE5C** VBYTE: B keyword value.
- **BE5D-BE5E** VENDA: E keyword value.
- **BE5F-BE60** VLENTH: L keyword value.
- **BE61** VSLOT: S keyword value.
- **BE62** VDRIV: D keyword value.
- **BE63-BE64** VPFLD: F keyword value.
- **BE65-BE66** VRECD: R keyword value.
- **BE67** VVOLM: V keyword value (ignored).
- **BE68-BE69** VLINE: O keyword value.
- **BE6A** VTYPE: T keyword value (in hex).
- **BE6B** VIOSLT: PR# or IN# slot number value.
Beneath Apple ProDOS Supplement

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

addr description/contents
C600 module starting address

* boot rom = apple disk controller
* for apple II, II+, and Ile.
* this code resides from $C600
* to $C6FF, it loads track 0
* sector 0 into ram at $800 and
* jumps to it

* zero page addresses **********

$026 sector buffer pointer
$028 slot number * 16 for index
$03C workbyte
$03D sector wanted
$040 track found
$041 track wanted

******** external addresses ********

$0100 system stack
$0300 auxiliary buffer
$0356 translate table
$0800 sectors to load
$0801 entry point
$0802 phases off
$0803 phases on
$0809 motor on
$080A drive select
$080C read data register
$080E set read mode
$08A8 monitor wait routine
$08F8 RTS

C600 ************** build read translate table **************

C600 signature
C602 initialize table value indicator
C606 store bit pattern
C609 shift pattern left one bit
C60A are there any two adjacent bits on?
C60C no, try another pattern >>C61E
C610 flip bits, pair of zero bits now single one bit
C612 high bit always on/take off bit we missed before
C614 --- >>C61E
C616 shift pattern right, must have only one bit on

C617 if more than one bit on, try another pattern >>C614
C619 found one, get table value
C61A and store it in table ($0356)
C61D increment table value indicator
C61E get next bit pattern, done yet
C61F no, go check it out >>C606

C621 ************ determine slot, turn drive on **************

C621 call a known RTS <<F58>
C624 get stack pointer
C625 get high byte of where we are ($180)
C628 times 16 to get slot
C62C save slot
C62E put in X REG for INDEX
C62F insure read mode ($0BE)
C635 select drive 1 ($06A)
C638 turn the motor on ($089)

C63B **************** recalibrate disk arm ****************

C63B prepare to step the arm 80 phases
C63D turn a phase off ($080)
C648 put counter in accumulator
C64A create a phase number (0-3)
C643 double it for proper index
C644 combine with slot for final index
C646 put index in X register
C647 turn a phase on ($081)
C64A delay about 20 microseconds
C64F decrement counter
C650 loop until all 80 are done >>C63D

C652 ************ initialization ************

C652 ---
C654 sector to find -> $00
C656 track to find -> $00
C65A main buffer pointer ($26) -> $8000
C65C clear the carry
C65D push status on stack

C65E ************ search for a valid header ************

C65E check data register ($08C)
C661 loop until data is valid >>C65E
C663 is it a $D5?
C665 no, try again >>C65E
C667 yes, check register again ($08C)
C66A loop until valid >>C667
C66C is it an $AA
Disk Controller Boot ROM -- Apple II/II+//IIe  NEXT OBJECT ADDR: C66E

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>C66E</td>
<td>NO, SEE IF IT'S A $95 &gt;&gt;C663</td>
</tr>
<tr>
<td>C670</td>
<td>YES, DELAY FOR REGISTER TO CLEAR</td>
</tr>
<tr>
<td>C671</td>
<td>CHECK REGISTER (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 $AD?</td>
</tr>
<tr>
<td>C67F</td>
<td>YES, WE FOUND A DATA HEADER &gt;&gt;C6A6</td>
</tr>
<tr>
<td>C681</td>
<td>NO, START OVER &gt;&gt;C65C</td>
</tr>
<tr>
<td>C683</td>
<td>************** DECODE ADDRESS FIELD ***********************</td>
</tr>
<tr>
<td>C685</td>
<td>INITIALIZE COUNTER</td>
</tr>
<tr>
<td>C686</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 X1X1X11</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 X1X1X11 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;C685</td>
</tr>
<tr>
<td>C699</td>
<td>KEEP THE STACK CLEAN</td>
</tr>
<tr>
<td>C69A</td>
<td>IS THIS SECTOR WE WANT?</td>
</tr>
<tr>
<td>C69C</td>
<td>NO, START OVER &gt;&gt;C65C</td>
</tr>
<tr>
<td>C69E</td>
<td>GET TRACK FOUND</td>
</tr>
<tr>
<td>C6A6</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>
<tr>
<td>C6A6</td>
<td>************** READ DATA FIELD *****************************</td>
</tr>
<tr>
<td>C6A8</td>
<td>---</td>
</tr>
<tr>
<td>C6AA</td>
<td>READ DATA REGISTER (C08C)</td>
</tr>
<tr>
<td>C6AD</td>
<td>LOOP UNTIL VALID &gt;&gt;C6AA</td>
</tr>
<tr>
<td>C6AF</td>
<td>EXCLUSIVE-OR WITH TRANSLATE TABLE (0206)</td>
</tr>
<tr>
<td>C6B4</td>
<td>DECREMENT OFFSET</td>
</tr>
<tr>
<td>C6B5</td>
<td>STORE BYTE IN AUXILIARY BUFFER (0308)</td>
</tr>
<tr>
<td>C6B8</td>
<td>LOOP UNTIL BUFFER FULL &gt;&gt;C6B8</td>
</tr>
<tr>
<td>C6BA</td>
<td>INITIALIZE OFFSET (MAIN BUFFER)</td>
</tr>
<tr>
<td>C6BC</td>
<td>READ DATA REGISTER (C08C)</td>
</tr>
<tr>
<td>C6BF</td>
<td>LOOP UNTIL VALID &gt;&gt;C6BC</td>
</tr>
<tr>
<td>C6C1</td>
<td>EXCLUSIVE-OR WITH TRANSLATE TABLE (0206)</td>
</tr>
<tr>
<td>C6C6</td>
<td>STORE BYTE IN MAIN BUFFER</td>
</tr>
<tr>
<td>C6C8</td>
<td>INCREMENT OFFSET</td>
</tr>
<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>
</tbody>
</table>
Disk Controller Boot ROM -- Apple IIc

Address: C552

Description/Contents

C552
Module Starting Address

****************************************************************************
* BOOT ROM - APPLE //C CONTROLLER ROM
* THIS CODE RESIDES FROM $C552 TO $C6FF. IT LOADS TRACK 0
* SECTOR 0 INTO RAM AT $8000 AND JUMPS TO IT. IF BOOT FAILS IT
* THEN TRIES TO BOOT SLOT 5,
* THE PROTOCOL CONVERTER.
* THIS IS THE VERSION OF THE IIc ROM
* THAT SUPPORTS THE UNIDISK 3.5,
* 26 JULY 85.
*****************************************************************************

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

$001  Slot Page Put Here During AutoBoot
$002  Retry Count (High Byte)
$026  Sector Buffer Pointer
$028  Slot Number * 16 For Index
$03C  WorkByte
$03D  Sector Wanted
$040  Track Found
$041  Track Wanted
$04F  Drive To Boot From

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

$030  Auxiliary Buffer
$036  Translate Table
$07B  Screen Location
$080  Sectors To Load
$081  Entry Point
$08B  PHASE0 OFF
$08C  PHASE0 ON
$088  MOTOR OFF
$089  MOTOR ON
$08C  Read Data Register
$08E  Set Read Mode
$09A  Drive Select
$0CA  Monitor Wait Routine

Disk Controller Boot ROM -- Apple IIc

Address: C552

Description/Contents

C552
Slot Page Put Here During AutoBoot

$002
Retry Count (High Byte)

$026
Sector Buffer Pointer

$028
Slot Number * 16 For Index

$03C
WorkByte

$03D
Sector Wanted

$040
Track Found

$041
Track Wanted

$04F
Drive To Boot From

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

$001
Slot Page Put Here During AutoBoot

$026
Sector Buffer Pointer

$028
Slot Number * 16 For Index

$03C
WorkByte

$03D
Sector Wanted

$040
Track Found

$041
Track Wanted

$04F
Drive To Boot From

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

$030
Auxiliary Buffer

$036
Translate Table

$07B
Screen Location

$080
Sectors To Load

$081
Entry Point

$08B
PHASE0 OFF

$08C
PHASE0 ON

$088
MOTOR OFF

$089
MOTOR ON

$08C
Read Data Register

$08E
Set Read Mode

$09A
Drive Select

$0CA
Monitor Wait Routine

********* SLOTS CODE ***********

The following two routines are in the $C500 area but are used by the $C600 logic.

C552
BootFail
Come here if boot fails. Put message on screen and go to sleep forever.

C552
17 Characters In Message

C557
Put At Bottom Of Screen ($7DB)

C55D
Then Go To Sleep >>C550

C55E
"Check Disk Drive"

C56F
Skip Over Miscellaneous Code

C56F
Slot 5 Logic In Here

C58E
Build Read Translate Table

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

C58E
Initialize Bit Pattern

C590
Initialize Table Value Indicator

C592
Store Bit Pattern

C593
Shift Pattern Left One Bit

C596
Are There Any Two Adjacent Bits On?

C598
No, Try Another Pattern >>C5AA

C59A
Yes, Turn Off Rightmost Of Each Group Of Zeros

C59C
Flip Bits. Pair Of Zero Bits Now Single Bit, Etc

C59E
High Bit Always On/Turn Off Bit We Missed Before

C5A0
--- >>C5AA

C5A2
Shift Pattern Right, Must Have Only One Bit On

C5A3
If More Than One Bit On, Try Another Pattern >>C5A0

C5A5
Found One, Get Table Value

C5A6
And Store It In Table ($356)

C5A9
Increment Table Value Indicator

C5AA
Get Next Bit Pattern, Done Yet?

C5A8
No, Go Check It Out >>C592

C5AD
Main Buffer Pointer ($26) -> $0800

C5B1
Initialize Retry Count (Low Byte)

C5B3
Return To Caller

C5B4
Skip Over Miscellaneous Code

C5B4
Slot 5 Logic In Here
Disk Controller Boot ROM -- Apple IIC

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>C5F5</td>
<td>JUMP TO BOOTFAIL</td>
</tr>
<tr>
<td>C5F5</td>
<td>BRANCH TO BOOTFAIL &gt;&gt;C52</td>
</tr>
<tr>
<td>C5F8</td>
<td>REMAINING 8 BYTES NOT USED BY DISK II &gt;&gt;C576</td>
</tr>
<tr>
<td>C600</td>
<td>SIGNATURE</td>
</tr>
<tr>
<td>C602</td>
<td>SET DRIVE -&gt; 1</td>
</tr>
<tr>
<td>C604</td>
<td>INITIALIZE RETRY COUNT (HIGH BYTE)</td>
</tr>
<tr>
<td>C608</td>
<td>SELECT DRIVE AND TURN IT ON</td>
</tr>
<tr>
<td>C60B</td>
<td>INITIALIZE SLOT (6)</td>
</tr>
<tr>
<td>C60D</td>
<td>INITIALIZE DEVICE (1 OR 2)</td>
</tr>
<tr>
<td>C60F</td>
<td>SAVE DRIVE NUMBER ON STACK</td>
</tr>
<tr>
<td>C610</td>
<td>INSURE READ MODE (C08E)</td>
</tr>
<tr>
<td>C616</td>
<td>GET DRIVE NUMBER BACK</td>
</tr>
<tr>
<td>C617</td>
<td>SELECT APPROPRIATE DRIVE (C08A)</td>
</tr>
<tr>
<td>C61A</td>
<td>TURN MOTOR ON (C089)</td>
</tr>
<tr>
<td>C61D</td>
<td>RECALIBRATE DISK ARM</td>
</tr>
<tr>
<td>C61F</td>
<td>PREPAIR TO STEP THE ARM 80 PHASES</td>
</tr>
<tr>
<td>C621</td>
<td>TURN A PHASE OFF (C080)</td>
</tr>
<tr>
<td>C622</td>
<td>PUT COUNTER IN A REGISTER</td>
</tr>
<tr>
<td>C623</td>
<td>CREATE A PHASE NUMBER (0-3)</td>
</tr>
<tr>
<td>C625</td>
<td>DOUBLE IT FOR PROPER INDEX</td>
</tr>
<tr>
<td>C626</td>
<td>COMBINE WITH SLOT FOR FINAL INDEX</td>
</tr>
<tr>
<td>C628</td>
<td>PUT INDEX IN X REGISTER</td>
</tr>
<tr>
<td>C629</td>
<td>TURN A PHASE ON (C081)</td>
</tr>
<tr>
<td>C62C</td>
<td>DELAY ABOUT 20 MICROSECONDS</td>
</tr>
<tr>
<td>C631</td>
<td>DECREMENT COUNTER</td>
</tr>
<tr>
<td>C632</td>
<td>LOOP UNTIL ALL 80 ARE DONE &gt;&gt;C61F</td>
</tr>
<tr>
<td>C634</td>
<td>INITIALIZATION</td>
</tr>
<tr>
<td>C636</td>
<td>SECTOR TO FIND -&gt; $00</td>
</tr>
<tr>
<td>C63B</td>
<td>TRACK TO FIND -&gt; $00</td>
</tr>
<tr>
<td>C63A</td>
<td>BUILD THE TRANSLATE TABLE &lt;C58E&gt;</td>
</tr>
<tr>
<td>C63D</td>
<td>COUNT RETRIES AND INDICATE ERROR IF BOOT FAILS ****</td>
</tr>
<tr>
<td>C63D</td>
<td>INITIALIZE RETRY COUNT</td>
</tr>
<tr>
<td>C63F</td>
<td>CLEAR THE CARRY</td>
</tr>
<tr>
<td>C640</td>
<td>PUSH STATUS ON STACK</td>
</tr>
<tr>
<td>C641</td>
<td>KEEP STACK CLEAN</td>
</tr>
<tr>
<td>C642</td>
<td>GET SLOT</td>
</tr>
</tbody>
</table>

Disk Controller Boot ROM -- Apple IIC

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>C644</td>
<td>DECREMENT RETRY COUNT, TRY AGAIN?</td>
</tr>
<tr>
<td>C646</td>
<td>YES, GO DO IT &gt;&gt;C656</td>
</tr>
<tr>
<td>C648</td>
<td>NO, TURN DRIVE OFF (C088)</td>
</tr>
<tr>
<td>C64B</td>
<td>AUTO BOOT FROM SLOT?</td>
</tr>
<tr>
<td>C64F</td>
<td>NO, FAIL NOW &gt;&gt;C5F5</td>
</tr>
<tr>
<td>C651</td>
<td>MAYBE SLOT 5 WILL TALK TO US &gt;&gt;C570</td>
</tr>
<tr>
<td>C654</td>
<td>TWO BYTES NOT USED &gt;&gt;$002</td>
</tr>
<tr>
<td>C656</td>
<td>---</td>
</tr>
<tr>
<td>C657</td>
<td>DECREMENT RETRY COUNT (LOW BYTE)</td>
</tr>
<tr>
<td>C658</td>
<td>IF NOT ZERO, TRY AGAIN &gt;&gt;C65E</td>
</tr>
<tr>
<td>C65A</td>
<td>IF SO, GO DECREMENT RETRY COUNT (HIGH BYTE) &gt;&gt;C641</td>
</tr>
<tr>
<td>C65C</td>
<td>SPACE FILLER TO POSITION CODE BELOW &gt;&gt;C63D</td>
</tr>
<tr>
<td>C65E</td>
<td>SEARCH FOR A VALID HEADER</td>
</tr>
<tr>
<td>C661</td>
<td>CHECK DATA REGISTER (C08C)</td>
</tr>
<tr>
<td>C663</td>
<td>IS IT A $D5?</td>
</tr>
<tr>
<td>C665</td>
<td>NO, TRY AGAIN &gt;&gt;C657</td>
</tr>
<tr>
<td>C667</td>
<td>YES, CHECK REGISTER AGAIN (C08C)</td>
</tr>
<tr>
<td>C66A</td>
<td>LOOP UNTIL VALID &gt;&gt;C667</td>
</tr>
<tr>
<td>C66C</td>
<td>IS IT AN $AA</td>
</tr>
<tr>
<td>C66E</td>
<td>NO, SEE IF ITS A $D5 &gt;&gt;C663</td>
</tr>
<tr>
<td>C670</td>
<td>YES, DELAY FOR REGISTER TO CLEAR</td>
</tr>
<tr>
<td>C671</td>
<td>CHECK REGISTER (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;C63F</td>
</tr>
<tr>
<td>C67D</td>
<td>WAS IT AN $AD7</td>
</tr>
<tr>
<td>C67F</td>
<td>YES, WE FOUND A DATA HEADER &gt;&gt;C66</td>
</tr>
<tr>
<td>C681</td>
<td>NO, START OVER &gt;&gt;C63F</td>
</tr>
</tbody>
</table>

Disk Controller Boot ROM -- Apple IIC

<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 X1X1X1</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;C68F</td>
</tr>
<tr>
<td>C694</td>
<td>COMBINE WITH PREVIOUS X1X1X1 AND X1X1X1</td>
</tr>
<tr>
<td>C696</td>
<td>DECREMENT COUNTER, DONE YET?</td>
</tr>
<tr>
<td>C697</td>
<td>NO, DO ANOTHER &gt;&gt;C685</td>
</tr>
<tr>
<td>C699</td>
<td>KEEP THE STACK CLEAN</td>
</tr>
<tr>
<td>C69A</td>
<td>IS THIS SECTOR WE WANT?</td>
</tr>
<tr>
<td>C69C</td>
<td>NO, START OVER &gt;&gt;C63F</td>
</tr>
<tr>
<td>C69E</td>
<td>GET TRACK FOUND</td>
</tr>
</tbody>
</table>
Disk Controller Boot ROM -- Apple Iic

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>C6A0</td>
<td>IS IT TRACK WE WANT?</td>
</tr>
<tr>
<td>C6A2</td>
<td>NO, START OVER &gt;&gt;C63F</td>
</tr>
<tr>
<td>C6A4</td>
<td>YES, INDICATE ADDRESS FOUND, GO LOOK FOR DATA FIELD &gt;&gt;C642</td>
</tr>
<tr>
<td>C6A6</td>
<td>********** READ DATA FIELD ***********************</td>
</tr>
<tr>
<td>C6A6</td>
<td>INITIALIZE OFFSET (AUXILIARY BUFFER)</td>
</tr>
<tr>
<td>C6A8</td>
<td>---</td>
</tr>
<tr>
<td>C6AA</td>
<td>READ DATA REGISTER (C08C)</td>
</tr>
<tr>
<td>C6AD</td>
<td>LOOP UNTIL VALID &gt;&gt;C66A</td>
</tr>
<tr>
<td>C6AF</td>
<td>EXCLUSIVE-OR WITH TRANSLATE TABLE (02D6)</td>
</tr>
<tr>
<td>C6B4</td>
<td>DECREMENT OFFSET</td>
</tr>
<tr>
<td>C6B5</td>
<td>STORE BYTE IN AUXILIARY BUFFER (0300)</td>
</tr>
<tr>
<td>C6B8</td>
<td>LOOP UNTIL BUFFER FULL &gt;&gt;C6A8</td>
</tr>
<tr>
<td>C6BA</td>
<td>INITIALIZE OFFSET (MAIN BUFFER)</td>
</tr>
<tr>
<td>C6BC</td>
<td>READ DATA REGISTER (C08C)</td>
</tr>
<tr>
<td>C6BF</td>
<td>LOOP UNTIL VALID &gt;&gt;C6BC</td>
</tr>
<tr>
<td>C6C1</td>
<td>EXCLUSIVE-OR WITH TRANSLATE TABLE (02D6)</td>
</tr>
<tr>
<td>C6C6</td>
<td>STORE BYTE IN MAIN BUFFER</td>
</tr>
<tr>
<td>C6C8</td>
<td>INCREMENT OFFSET</td>
</tr>
<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? (02D6)</td>
</tr>
<tr>
<td>C6D3</td>
<td>NO, START OVER &gt;&gt;C6A2</td>
</tr>
<tr>
<td>C6D5</td>
<td>********** MERGE MAIN AND AUXILIARY BUFFERS**************</td>
</tr>
<tr>
<td>C6D5</td>
<td>INITIALIZE OFFSET (MAIN BUFFER)</td>
</tr>
<tr>
<td>C6D7</td>
<td>INITIALIZE OFFSET (AUXILIARY BUFFER)</td>
</tr>
<tr>
<td>C6D9</td>
<td>DECREMENT OFFSET (AUX BUFFER)</td>
</tr>
<tr>
<td>C6DA</td>
<td>IF LESS THAN ZERO RESET IT &gt;&gt;C6D7</td>
</tr>
<tr>
<td>C6DC</td>
<td>GET BYTE FROM MAIN BUFFER</td>
</tr>
<tr>
<td>C6E1</td>
<td>ROLL IN TWO BITS FROM AUXILIARY BUFFER</td>
</tr>
<tr>
<td>C6E6</td>
<td>SAVE COMPLETED DATA BYTE</td>
</tr>
<tr>
<td>C6E8</td>
<td>INCREMENT OFFSET (MAIN BUFFER)</td>
</tr>
<tr>
<td>C6E9</td>
<td>LOOP UNTIL WHOLE BUFFER IS DONE &gt;&gt;C6D9</td>
</tr>
<tr>
<td>C6EB</td>
<td>********** DETERMINE IF THERE IS MORE TO DO***************</td>
</tr>
<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? (0808)</td>
</tr>
<tr>
<td>C6F6</td>
<td>YES, GO DO IT &gt;&gt;C6D3</td>
</tr>
<tr>
<td>C6F8</td>
<td>NO, ENTER CODE WE JUST LOADED &gt;&gt;0801</td>
</tr>
<tr>
<td>C6FB</td>
<td>5 ZERO BYTES AT END OF PAGE</td>
</tr>
</tbody>
</table>
ERRATA TO BENEATH APPLE PRODOS (1st Printing, 1984)

You can identify which printing of Beneath Apple ProDOS you have by looking at the space between the title of the book and the author's names on the first page of the book (the title page). If this space is blank, you have the first printing. The second printing has "Second Printing, March 1985" in this space. If you have the second printing, skip to page 120. If you have the first printing, all of the following errata apply.

Page 3-16:
In the first paragraph starting on the page, the sentence should read "The data is dealt with in larger pieces (512 bytes vs. 256 bytes)...", not 512K vs. 256K.

Page 6-63:
The code for "HOW MUCH MEMORY IS IN THIS MACHINE?" is incorrect. Replace it with:

```
LDA $BF98   GET MACHID FROM GLOBAL PAGE
ASL A      MOVE BITS TO TEST POSITION
ASL A
BPL SMLMEM 48K
ASL A
BVS MEM128 128K
      OTHERWISE 64K
```

Page 6-64:
The code for "GIVEN A PAGE NUMBER, SEE IF IT IS FREE" is incorrect. Replace it with:

```
BITMAP EQU $BF58 SEE PAGE 8-6
LDA #PAGE GET PAGE NUMBER (MSB OF ADDR)
JSR LOCATE LOCATE ITS BIT IN BITMAP
AND BITMAP,Y IS IT ALLOCATED?
BNE INUSE YES, CAN'T TOUCH IT
TXA
ORA BITMAP,Y PUT BIT PATTERN IN ACCUM
STA BITMAP,Y MARK THIS PAGE AS IN USE
      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 DIVIDE PAGE BY 8
LSR A
TAY Y-REG IS OFFSET INTO BITMAP
TXA PUT BIT PATTERN IN ACCUM
RTS DONE

BITMASK DFB $80,$40,$20,$10 BIT MASK PATTERNS
DFB $08,$04,$02,$01

Page 7-9

The code on page 7-9 is incorrect and should be replaced with the following:

* SQUISH OUT DEVICE NUMBER FROM DEVLST
SKP 1
LDX $BF31 GET DEVCNT
DEVLP LDA $BF32,X PICK UP LAST DEVICE NUM
AND #$70 ISOLATE SLOT
CMP #$30 SLOT = 3?
BEQ GOTS LT YES, CONTINUE
DEX
BPL DEVLP CONTINUE SEARCH BACKWARDS
BMI NORAM CAN'T FIND IT IN DEVLST
GOTS LT LDA $BF32+1,X GET NEXT NUMBER
STA $BF32,X AND MOVE THEM FORWARD
INX
CPX $BF31 REACHED LAST ENTRY?
BNE GOTS LT NO, LOOP
DEC $BF31 REDUCE DEVCNT BY 1
LDA #$0 ZERO LAST ENTRY IN TABLE
STA $BF32,X
CLC
BCC OKXIT BRANCH ALWAYS TAKEN
SKP 1
OLDVEC DW 0 OLD VECTOR SAVEAREA
To reinstall the /RAM driver, execute this subroutine:

```
SKP  1
SEE IF SLOT 3 HAS A DRIVER ALREADY
SKP  1
HIMEM EQU $73 PTR TO BI'S GENERAL PURPOSE BUFFER
SKP  1
INSTALL LDX $BF31 GET DEVCNT
INSLP LDA $BF32,X GET A DEVNUM
AND #$70 ISOLATE SLOT
CMP #$30 SLOT 3?
BEQ INSOUT YES, SKIP IT
DEX
BPL INSLP KEEP UP THE SEARCH
SKP  1
* RESTORE THE DEVNUM TO THE LST
SKP  1
LDX $BF31 GET DEVCNT AGAIN
CPX #$0D DEVICE TABLE FULL?
BNE INSLP2
ERROR ...
INSLP2 LDA $BF32-1,X MOVE ALL ENTRIES DOWN
STA $BF32,X TO MAKE ROOM AT FRONT
DEX BNE INSLP2
LDA #$08 FOR A NEW ENTRY
STA $BF32 SLOT 3, DRIVE 2 AT TOP OF LIST
INC $BF31 UPDATE DEVCNT
SKP  1
```

Page 7-26:

Modifying the ProDOS Disk II Device Driver to allow 320 blocks instead of the normal 280. The fourth command line should read:

```
520D:40
```

Modifying FILER to format 40 tracks instead of 35. The fourth command line should read:

```
4244:40
```

[See Second printing errata for information about versions other than 1.0.1]
Page 8-6:

Under "device Information", make the following changes:

BF10-BF11 DEVADR01 Slot 0 reserved.

...  BF26-BF27 DEVADR32 /RAM device driver address (need extra 64K).

Page 8-7:

The wrong bit is indicated as the "expansion bit" in the MACHID byte. The first eight rows of that description should read:

00.. 0... II
01.. 0... II+
10.. 0... IIe
11.. 0... III emulation
00.. 1... Future expansion
01.. 1... Future expansion
10.. 1... IIC
11.. 1... Future expansion

Page B-8:

In the last paragraph, the sentence should read "A second way to use an interpreted language..." (not a compiled language).

Page D-1:

In the second paragraph, the sentence should read "Versions of the Disk Drive Controller Unit are now used..." (not based).

Reference Card, Panel 4

Under "SYSTEM GLOBAL PAGE FORMAT", replace the lines beginning BF05 and BF06 with the following two lines:

BF06 Jump to Date/Time Address (or RTS if no clock)
The description of BF10-11 should be changed to:

BF10-11 Slot 0 reserved

The description of BF26-27 should be changed to:

BF26-27 /RAM

Under the "MACHINE IDENTIFICATION BYTE", the second column of numbers should read:

0...
0...
0...
0...
1...
1...
1...
1...

Reference Card, Panel 9

The last entry for "MLI ERROR CODES" should be:

$5A  Bad vol. bit map

(not $58).
ERRATA TO BENEATH APPLE PRODOS (2nd Printing, 1985)

Page 4-30

The definitions of PARENT POINTER and PARENT ENTRY are incorrect. Replace them with:

$27-$28 PARENT_POINTER: The block number (within the volume directory or a subdirectory) which contains the file entry for this subdirectory.

$29 PARENT_ENTRY: The number of the file entry within the block number pointed to by the PARENT_POINTER. Given that "ENTRIES_PER_BLOCK" is $0D, then the PARENT_ENTRY number ranges from $01 to $0D.

Page 7-26

Expand the 40-track drive patch to show how to patch PRODOS versions 1.0.2 and 1.1.1 as well as 1.0.1.

This patch modifies the Disk II Driver, which is a part of the "PRODOS" file, so that it allows 320 blocks per volume instead of 280 blocks per volume.

UNLOCK PRODOS
BLOAD PRODOS,TSYS,$A2000
CALL -151
address*:40
3D0G
BSAVE PRODOS,TSYS,$A2000
LOCK PRODOS

*"address" varies with the version of ProDOS, as follows:

<table>
<thead>
<tr>
<th>ProDOS Version</th>
<th>address</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0.1</td>
<td>520D</td>
</tr>
<tr>
<td>1.0.2</td>
<td>52CD</td>
</tr>
<tr>
<td>1.1.1</td>
<td>56E3</td>
</tr>
</tbody>
</table>
The following patch modifies the program FILER to format 40 tracks instead of 35. After this modification is made, only 40-track drives may be formatted with FILER.

UNLOCK FILER
BLOAD FILER, TSYS, A$2000
CALL -151
addr**: 40
79F4: 28
3D0G
BSAVE FILER, TSYS, A$2000
LOCK FILER

**"addr" depends on the release date of FILER. Here are the values of "addr" for two different release dates:

<table>
<thead>
<tr>
<th>Release date</th>
<th>addr</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 JAN 84</td>
<td>4244</td>
</tr>
<tr>
<td>18 JUN 84</td>
<td>426A</td>
</tr>
</tbody>
</table>
Quality Software Products For the Apple

BOOKS

Beneath Apple ProDOS by Don Worth & Pieter Lechner
Describes the ProDOS Operating System clearly and in detail, going beyond Apple's manuals. Many programming examples are included. 288 pages. 176 pages. $19.95
Supplements to Beneath Apple ProDOS:
Versions 1.0.1 and 1.0.2 (combined) $10.00
Version 1.1.1 $12.50

Beneath Apple DOS by Don Worth & Pieter Lechner
The popular best seller that covers all facets of DOS 3.3 and previous Apple disk operating systems. 176 pages. $19.95

Understanding the Apple II by Jim Sather
Foreword by Steve Wozniak. A definitive source of information, covers Apple II and Apple II Plus hardware, including the disk controller and logic state sequencer. 352 pages. $22.95

Understanding the Apple IIe by Jim Sather
The companion to Understanding the Apple II, this book covers Apple IIe hardware, including video graphics and the 1985 firmware upgrade (65C02). 368 pages. $24.95

UTILITIES

Bag of Tricks 2 by Don Worth & Pieter Lechner
Quality Software's popular set of Apple II disk utility programs, Bag of Tricks, has been thoroughly revised and updated for the ProDOS operating system. TRAX, INIT, ZAP, and FIXCAT are the four comprehensive utility programs, all with improved user interfaces to make them easier to use than the original Bag of Tricks.* Unprotected diskette and 200-page manual. 64K. $49.95

*Special offer to Bag of Tricks owners--save $20 by ordering directly from Quality Software. To order, send in your Bag of Tricks diskette and $29.95, plus shipping, handling, and sales tax. We will return your diskette along with the new product.

Universal File Conversion by Gary Charpentier
Moves programs and data among the five operating systems used on the Apple II family of computers: DOS, ProDOS, CP/M, Pascal, and SOS. Unprotected diskette and 48-page manual. 64K. $34.95
Ordering directly from Quality Software

To order our products directly, mail this order form to Quality Software (at the address below) with your payment—the price of the software (plus sales tax if shipped to California) plus shipping and handling charges. Your payment can be a check or bank draft made payable to Quality Software in US dollars, or your VISA or MASTERCARD number and expiration date (VISA and MASTERCARD holders may phone in their orders). California residents must add the appropriate sales tax (6%, 6.5%, or 7%).

Shipping charges:
- 48 Continental United States (UPS) ............... $2.50
- Alaska, Hawaii, Canada, and Mexico (air mail) .... $5.00
- All other countries (insured air mail) ............. $10.00

Send your order to:

QUALITY SOFTWARE
21610 Lassen Street #7
Chatsworth CA 91311
(818) 709-1721

<table>
<thead>
<tr>
<th>QUANTITY</th>
<th>DESCRIPTION</th>
<th>AMOUNT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

SUBTOTAL
(CA RESIDENTS) SALES TAX
SHIPPING
TOTAL

Check # __________
OR VISA/MasterCard # ___________________ EXPIRES ______

Name ________________________________
Street Address ________________________
City, State, Postal Code ____________________
Country ______________________________

(1.1.1)
SUPPLEMENT TO

Beneath Apple ProDOS

For ProDOS Version 1.1.1

by Don Worth and Pieter Lechner

QS QUALITY SOFTWARE