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

$19.95

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

$10.00

Beneath Apple DOS
by Don Worth & Pieter Lechner

$19.95

Understanding the Apple II
by Jim Sather

$22.95

Understanding the Apple IIe
by Jim Sather

$24.95

Apple Utility Software from Quality Software

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

$49.95

Universal File Conversion (includes diskette)
by Gary Charpentier

$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.
<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>
A ProDOS ATLAS
(such as zero-page usage, etc.) 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
   I plus one index I
   I block) I
   I L$3A00 I
   I I
   I 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

MODULE STARTING ADDRESS

* PRODOS LOADER
* THIS CODE IS LOADED FROM BLOCK 0
* INTO MEMORY AT $0000.
* ITS PURPOSE IS TO LOAD THE "PRODOS"
* FILE INTO $2000 AND JUMP TO IT.
* (PRODOS RELOCATOR IS AT $2000).
* VERSION 1.1.1 -- 18 SEP 84
* (THE LOADER IS STILL THE SAME AS IT
* WAS IN VERSION 1.0.1)

*** EXTERNAL ADDRESSES ***

ROM BOOT SUBRTN BUFFER PAGE ADDR
ROM BOOT SUBRTN SLOT # 16
ROM BOOT SUBRTN SECTOR TO READ
ROM BOOT SUBRTN CURRENT TRACK
ROM SUBRTN TRACK TO READ
-- BLOCK READ PARAMETER LIST --
COMMAND NUMBER (1 = READ)
SLOT # 16
I/O BUFFER ADDRESS ($44/$45)
BLOCK TO READ ($46/$47)

POINTER TO BLOCK READ ROUTINE
VOL DIR ENTRY POINTER/FIRST INDEX PAGE
ADDR OF SECOND PAGE OF INDEX BLOCK
INDEX INTO INDEX BLOCK PAGES
TRACK SEEK PHASE-ON INDEX
TRACK PHASE WANTED
BLOCK READER RETRY COUNT
CURRENT TRACK PHASE/PHASE-OFF INDEX
BUFFER POINTER
SCREEN CENTER LINE
LOAD POINT FOR RELOCATOR
DISK ARM PHASE0
TURN DISK DRIVE OFF
TURN DISK DRIVE ON
SHIFT DATA REGISTER

PC59
HOME CURSOR/CLEAR SCREEN

0000
SIGNATURE BYTE ($01 MEANS BOOT ROUTINE FOLLOWS)
(A $03 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
JUMPS TO $A000. WHAT IS SHOWN HERE AS
$000 ON AN APPLE II IS $A000 ON AN APPLE ///.
THUS AN APPLE /// EXECUTES A HARMLESS
INSTRUCTION (ORA $30,X), THEN DOES NOT BRANCH
ON CARRY, AND JUMPS TO $A132 ($932 ON AN
APPLE II). MANY THANKS TO DAV HOLLE FOR
PROVIDING US WITH THIS APPLE /// INFORMATION.

0001
MAIN ENTRY

ON ENTRY, X = SLOT*16
A = SECTOR NUMBER

0002
ENTRY POINT FOR APPLE II
0002
ALWAYS TAKEN (APPLE II) >>0007
0004
JUMP TO APPLE /// LOGIC >>A132
0007
SAVE SLOT*16
0009
READING SECTOR 3 NEXT?
000B
REMEMBER THIS...
000D
MARK $CX FROM SLOT*16
0015
AND SAVE AT $49
0019
$4B/49 --> $CXFF IN ROM BOOT
001C
CHECK $CXFF
001D
BOOT ROM FOR DISK 117
001F
NO, NOT A 5.25" FLOPPY >>05B
0021
GOT BOTH SECTORS OF LOADER? >>0631
0023
NO, STOP AT SECTOR 3
0025
STORE ON PARM ($006)
0028
SKIP SECTOR 1 (GET SEC 2)
002A
DUMMY UP $CXXC AS RETURN ADDRESS
0030
AND CALL ROM SECTOR READ SUBRTN

****** LOAD PRODOS ******

(ENTIRE LOADER IN MEMORY NOW)

0031
CURRENT TRACK IS ZERO
0033
$4B/49 --> $CX00
0037
COPY A PORTION OF DISKETTE BOOT ROM
0039
TO MY BLOCK READER SUBRTN ($994)
003D
FROM $9F7 TO $A7E
0043
MODIFY SOME BRANCHES IN THE COPIED CODE ($91D)
0046
TO SUIT MY ERROR HANDLING TASTES ($924)
004C
AND COPY SECTOR READ SUBRTN EXIT CODE ($928)
ProDOS Loader -- V1.1 -- 18 SEP 84

`84F

PRODOS LOADER -- V1.1

84F TO $A7F TO $A85 ($A7F)

055  $48/49 --> DISKETTE BLOCK READER SUBRTN

059  AT $9986

059  ---

05D  LEGAL DISK ROM?

05F  NO, ERROR >>0890

061  STORE LSB OF BLOCK READER

063  STORE ZERO'S IN SEVERAL THINGS

065  COMMAND = 1 (READ BLOCK)

071  BLOCK NUMBER = 2 (Vol Directory)

075  $6B/61 --> $C00 (BUFFER)

077  $4A/4B --> $C00 (FIRST ENTRY)

079  READ VOLUME DIRECTORY BLOCKS <0912>

07C  ERROR? >>0866

07E  MOVE UP TWO PAGES IN MEMORY

082  NEXT BLOCK NUMBER

086  NOW AT BLOCK 6?

088  NO, GO READ NEXT ONE >>08279

08A  YES, CHECK LINK FOR VALIDITY ($C00)

08D  IT SHOULD BE ZERO FOR VOLT DIR ($C01)

08F  BAD VOLUME DIR IF NOT ZERO >>089F

092  NO, INDEX PAST LINK AND VOL HDR

094  AND BEGIN >>0898

096  IF ALREADY PROCESSING, USE ENTRY LSB

098  ----

099  ADD ENTRY LENGTH TO FIND NEXT ENTRY ($C23)

09D  STILL IN SAME PAGE? >>08AC

09F  NO, BUMP ENTRY MSB

0AF  IS IT ODD? (SECOND PAGE OF A BLOCK?)

0AC  YES... >>08AC

0AE  NO, JUST FINISHED LAST BLOCK?

0AB  YES, ERROR >> FILE NOT FOUND >>08FF

0A4  ELSE, START JUST PAST LINKS

0AC  UPDATE LSB OF ENTRY POINTER

0AE  GET NAME LENGTH ($902)

0B1  MASK OFF STORAGE TYPE

0B4  COMPARE NAME WITH "PRODOS"

0BB  NOT A MATCH? >>0996

0BE  IF NAME MATCHES, IS IT A SAPLING FILE?

0C2  IF NOT, I CAN'T HANDLE IT >>08FF

0C5  GET FILE TYPE

0CB  SHOULD BE A PRODOS SYS FILE

0CA  IF NOT, I GIVE UP >>08FF

0CD  ALL IS WELL, COPY KEY BLOCK NUMBER

0CF  TO $46/47

0D6  $4A/4B AND $6B/61 --> $1E00

0DB  (BUFFER TO HOLD KEY BLOCK)

0DB  $4C/4D --> $1F00 (SECOND PAGE)

0E3  READ A BLOCK <0912>

0E5  ERROR? >>08FF

0EA  BUMP TO NEXT BLOCK BUFFER

ProDOS Loader -- V1.1

---

`8EE

$4E = OFFSET INTENTS

$8FF GET NEXT BLOCK

$8F0 BLOCK NUMBER

$8F0A NOT YET, READ INTO INDEX BLOCK

$8F0C ELSE, JUMP TO NUMBER FROM INDEX BLOCK

$8F0D = 07 (END OF FILE)

$8F0F ERROR JUMP >> A BLOCK >> O83E

$8F01 RELOCATOR AT $2000 >> 2000

0902 ERROK

0902 LENGTH OF KERN

0903 'PRODOS' FILE NAME

0912 COPYHEL'S NAME

0912 COPY $6B/61 --

0914 (BLOCK READ P BLOCK READ BUFFER PTR

0917 A THEN GO TO HIC

091D ROM OFFSET POINTER

091D OF OFFSET 1/0 ROUTINE >>004B

091D TO BRANCH

091D BE CHANGEOCTORE READ OFFSETS

091D INTO ROM SECTOR READ SUBROUTINE

091D DISPLACEMENTS WHICH NEED TO

091D FOR LOADER'S PURPOSES

0924 ---

0924 --- IN BRANCH OFFSETS FOR ABOVE ---

092B SECTION

092B COPIED TO

092B GET SLOT*16 OR READ EXIT CODE

092D AND EXIT HERE END OF DISKETTE SECTOR READ CODE

092E RETURN

092F RESTART BLOCK

0932 APPL

0932 READ OPERATION >>098C

0A32 THIS IS $A32

0932 MAKE IT LOOK 3: /// BOOT CODE

093A LOAD IN BLOCK

093C GO TO APPLE /// WHEN BOOED ON APPPE ///

093F ERROR (WE WANT S0S, NOT PRODOS)

0947 BLOCK READ ROUTINE >>0979

0948 HANDLER
093F HOME CURSOR/CLEAR SCREEN <FC5B>
0944 COPY "UNABLE TO LOAD PRODOS" MESSAGE (0950)
0947 TO SCREEN (05AE)
094D THEN GO TO SLEEP FOREVER >>094D
0950 *** UNABLE TO LOAD PRODOS ***

096D ********** 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
096J 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 <098C>
09A5 NEXT PAGE
09B1 SKEW TO NEXT SECTOR
09B5 READ SECOND SECTOR OF BLOCK <098C>
09B8 THEN TURN MOTOR OFF AND EXIT (C086)
09BB 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...
<table>
<thead>
<tr>
<th>Address</th>
<th>Description/Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>MODULE STARTING ADDRESS</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>0000</td>
<td>PRODOS RELOCATOR</td>
</tr>
<tr>
<td>0001</td>
<td>LOADED AS THE FIRST</td>
</tr>
<tr>
<td>0002</td>
<td>PORTION OF THE PRODOS</td>
</tr>
<tr>
<td>0003</td>
<td>IMAGE AT $2000</td>
</tr>
<tr>
<td>0004</td>
<td>VERSION 1.1.1 -- 18 SEP 84</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address</th>
<th>Screen Line Addresses</th>
</tr>
</thead>
<tbody>
<tr>
<td>04B8</td>
<td>SCREEN BUFFER LINE</td>
</tr>
<tr>
<td>05A9</td>
<td>SCREEN BUFFER LINE</td>
</tr>
<tr>
<td>05B1</td>
<td>SCREEN BUFFER LINE</td>
</tr>
<tr>
<td>06A8</td>
<td>SCREEN BUFFER LINE</td>
</tr>
<tr>
<td>07A5</td>
<td>SCREEN BUFFER LINE</td>
</tr>
<tr>
<td>07D0</td>
<td>SCREEN BUFFER LINE</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address</th>
<th>Interp Loader Addresses</th>
</tr>
</thead>
<tbody>
<tr>
<td>0B0</td>
<td>ENTRY OF INTERP LOADER</td>
</tr>
<tr>
<td>0BE2</td>
<td>'UNABLE TO FIND SYSTEM FILE'</td>
</tr>
<tr>
<td>09A0</td>
<td>'INTERP FILE TOO LARGE'</td>
</tr>
<tr>
<td>092A</td>
<td>'UNABLE TO LOAD ...'</td>
</tr>
<tr>
<td>093B</td>
<td>INTERP FILE NAME ITSELF</td>
</tr>
<tr>
<td>093C</td>
<td>+1</td>
</tr>
<tr>
<td>094F</td>
<td>LENGTH OF MESSAGE</td>
</tr>
<tr>
<td>0951</td>
<td>ML: OPEN LIST</td>
</tr>
<tr>
<td>0956</td>
<td>ML: GET EOF</td>
</tr>
<tr>
<td>0958</td>
<td>EOF MARK</td>
</tr>
<tr>
<td>0959</td>
<td>EOF MARK+1</td>
</tr>
<tr>
<td>095A</td>
<td>EOF MARK+2 (MSB)</td>
</tr>
<tr>
<td>095B</td>
<td>ML: READ LIST</td>
</tr>
<tr>
<td>0960</td>
<td>READ BUFFER ADDR</td>
</tr>
<tr>
<td>0963</td>
<td>+1</td>
</tr>
<tr>
<td>0963</td>
<td>ML: CLOSE LIST</td>
</tr>
<tr>
<td>0965</td>
<td>'.SYSTEM'</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address</th>
<th>Miscellaneous Addresses</th>
</tr>
</thead>
<tbody>
<tr>
<td>0C0</td>
<td>VOLUME DIRECTORY BUFFER</td>
</tr>
<tr>
<td>0C23</td>
<td>ENTRY LENGTH</td>
</tr>
<tr>
<td>0044</td>
<td>RDMV DRIVE VOLUME DIRECTORY --</td>
</tr>
<tr>
<td>0045</td>
<td>VOLUME HDR, VOLUME NAME</td>
</tr>
<tr>
<td>0222</td>
<td>VOLUME HDR, ACCESS-TOTAL BLOCKS</td>
</tr>
<tr>
<td>0700</td>
<td>RAMDRIVE DEVICE DRIVER LOAD ADDRESS</td>
</tr>
<tr>
<td>0009</td>
<td>DIFFERENCE OF RAMDRIVE LOAD AND RUN LOCATIONS</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address</th>
<th>System Global Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>0F00</td>
<td>ENTRY POINT FOR MLI</td>
</tr>
<tr>
<td>0F01</td>
<td>QUIT VECTOR</td>
</tr>
<tr>
<td>0F06</td>
<td>DATE/TIME</td>
</tr>
<tr>
<td>0F10</td>
<td>DEVICE HANDLER TABLES</td>
</tr>
<tr>
<td>0F30</td>
<td>LAST DEVICE USED</td>
</tr>
<tr>
<td>0F31</td>
<td>NUMBER OF ACTIVE DISK DEVICES</td>
</tr>
<tr>
<td>0F32</td>
<td>ACTIVE DISKS SEARCH LIST</td>
</tr>
<tr>
<td>0F98</td>
<td>MACHINE TYPE FLAGS</td>
</tr>
</tbody>
</table>
BP99  SLC  TOXEM -- V1.1.1 -- 18 SEP 84  NEXT OBJECT ADDR: 2000

BPFP  TOXEM -- V1.1.1 -- 18 SEP 84  NEXT OBJECT ADDR: 2000

**OPTION/CONTENTS**

C000  80  WHICH CONTAIN CARDS WITH ROM
C002  80  48K RAM
C003  80  16K RAM
C004  WR  ******** I/O PORT ADDRESSES **********
C005  WR
C008  MA  STORE OFF
C009  AL  STORE ON
C00A  IN  MAIN RAM
C00B  PB  AUX RAM
C00C  80  MAIN RAM
C00D  80  AUX RAM
C018  WR  STACK/ZERO PAGE
C019  WR  STACK/.ZERO PAGE
C054  US  INTERNAL SLOT 3 ROM
C055  US  INTERNAL SLOT 3 ROM
C081  WR  COLUMN DISPLAY OFF
C082  MO  COLUMN DISPLAY ON
C083  RE  80STORE SWITCH
C088  RE  80STORE SWITCH
C311  MO  KEYBOARD ROM READ ENABLE
C314  TX  WRITE RAM 2ND 4K BANK
C314  TX  WRITE RAM 1ST 4K BANK

**INTERNAL C3ROM ADDRESSES**

C305  SL  TO/FROM AUXMEM SUBROUTINE
C30B  SL  TO/FROM AUXMEM SUBROUTINE
C30C  SL  TO/FROM AUXMEM SUBROUTINE
C30F  SL  TO/FROM AUXMEM SUBROUTINE

**SLOT ROM ADDRESSES**

C003  I.D. BYTE
C005  I.D. BYTE
C007  I.D. BYTE
C009  I.D. BYTE

D008  ST  I.D. BYTE
D013  EN  I.D. BYTE

R0  /O CARD ROMS

**PRODOS ADDRESSES**

8K OF QUICKE CODE MEMORY AREA (BANK2)

B00  RCVD ROM FLAG

C000  WR  CALLER ADDRESS

**MONITOR ROM**

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

2000  STORE SLOT IN MIL ONLINE PARMS
2005  PRINT "APPLE II PRODOS..." <2499>
2006  SET UP FOR COMMAND MOVES (228A)
2007  RELOCATE MIL SUBPROGRAMS & DATA TO LOW MEMORY <26A6>
2011  ERROR? >2450
2013  NO, PRODOS...
2017  BE SURE 48K OF MAIN MEMORY EXISTS (BPFP)
2018  IF NOT, ERROR >24DE
2026  MAKE DOUBLY SURE >206E
2028  SELECT FOTHERBOARD ROMS (C002)
2028  DETERMINE MACHINE TYPE >2492
2030  PICK UP CONFIGURATION BYTE
2032  64K OR M03 MEMORY?
2034  YES, WE HAVE 64K RAM >2039
2036  ERROR. MUST HAVE 64K FOR PRODOS 1.1.111 >21C3

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

2039  SET UP FOR M03 MOVE <228C>
2041  ERROR, >20BC
2044  ENABLE M03 BOARD ROMS AGAIN (C002)
2047  CHECK ROM I.D. BYTE (BPB3)
2048  APPLE //e FAMILY?
204C  NO, LEAVE I.D. BYTE AS IS >246D
2050  TEST ANOTHER ROM I.D. BYTE (FPBC0)
2053  SAVE BIT TEST RESULTS
2054  GET MACHID
2056  STRIP Bits THAT IDENTIFY MODEL
205b  IT'S A //e IF BITS 6 & 7 ARE HIGH >2069
205D  ---
205E  EITHER A //C OR A FUTURE SYSTEM
206b  CHECK HIGH BITS OF SPFBCO AGAIN
2061  BIT 7 ON >>246C
2062  YES, PUT IN SYSTEM...
2067  BIT 5 ON >>2068
2069  ---
206b  REPLACE UPDATED MACHID
206D  COPY BOX D DEVICES TO READ BLOCK PARMS (21FE)
AND AS LAST DEVICE USED (BF38)
Determine Peripheral Card Configuration <252A>
BOOT DEVICE TO... (2205)
GLOBAL PAGE LAST DEVICE USED (BF3B)
WRITE ENABLE BANK1 OF HIGH RAM (0C9B)
COPY CLOCK CODE TO DEVICE DRIVER AREA <26A5>
ERROR? >>2BF
CHECK MACHINE TYPE AGAIN (BF9C)
GOT 64K OR MORE?
NO >>20C2
YES, QUIT VECTOR -- SFE5
WRITE TO HIGH RAM (BANK2) (C0B3)
POINT TO QUIT CODE TABLE (2211)
MOVE QUIT CODE TO HIGH RAM <26A6>
STORE QUIT VECTOR START PAGE (0000)
WRITE TO HIGH RAM (BANK1) (C0B8)
AGAIN (C0B8)
RELOCATION ERROR >>21C3
GET MACHID YET AGAIN (BF9B)
128K?
YES, ESTABLISH RAM DRIVE IN UPPER 64K <28FF>
SET UP FOR IRQ (ENHANCED ROM) **
READ ROM (C0B1)
GET IRQ VECTOR FROM ROM (FFFE)
CARRY CLEAN IP IRQ VECTOR IN C3 ROM
IT'S AN OLD ROM >>20FD
READ & WRITE RAM (BANK1) (C0B8)
SWITCH TO AUX HIGH RAM (C0B9)
PUT IRQ VECTOR IN AUX HIGH RAM (FFFE)
BACK TO MAIN HIGH RAM, 2-PAGE (C0B8)
PUT IRQ VECTOR IN MAIN HIGH RAM (FFFF)
SET FLAG INDICATING
ENHANCED IRQ LOGIC ON BOARD (BFDB)
LOOK FOR SLOT 3 VIDEO CARD *****
ENABLE INTERNAL VIDEO FIRMWARE (C0BA)
CHECK FOR ROM (BF99)
IN SLOT 3.
NONE THERE >>216D
LOOK AT THE SLOT 3 ROM (C0B8)
THE MUST BE A $38
AND AT OFFSET +$97 (C307)
MUST BE AN $16
AND AT OFFSET +$9B (C30B)
MUST BE A 1
AND AT OFFSET +$9C (C30C)
INDICATED -- V1.1.1 -- 18 SEP 84 NEXT OBJECT ADDR: 2124
CHECK FUNCTION/CONTENTS
IS THIS...?
OTHER "H" AN 80-COL CARD.
MUST MACHINE TYPE (BF96)
GOOD, SO AN APPLE III?
GIVE US THE 80-COL CAPABILITY >>2165
TURN MANUFACTURERS MUST FOLLOW THE RULES! (C3FA)
HAVE BIT INSTRUCTION AT $C3FA
PUT A, YOU FOLLOWED THE RULES! >>2165
THE CONTROL BACK TO MOTHERBOARD ROM (C00A)
AND 80-COL (C0B1)
DON'T OR AUX MEM. (C055)
NO, YTE AT AUX $400 (0040)
SHIFT SIMULATOR LEFT
THE SAME WITH $400 (0040)
BACK THE SAME? (0040)
TURN THE 80-COL MEMORY >>2156
WAS SPO OF THE RIGHT
NO, SO WE SAME? (0040)
IN MAIN MEMORY (C054)
ALWAYS P 80-COL (C000)
TURN COL MEMORY FOUND? >>2165
TURN OFF 80-COL FLAG (BF9B)
THE I.D. BYTE:
BRANCH >>216A
MLI: ON 80-COL FLAG (BF9E)
ERRORS
VALID *** GET VOL LABEL ************
IF NOT
ELSE, BLINE DEVICE CALL <BF06>
AND F255 >>21C3
MLI: SEEK NAME?
ERROR: ERROR >>21C3
JUMP LENGTH BY ONE
NAME BY A "/"
T PREFIX BY A "/"

$14/15
** READ VOLUME DIRECTORY ************
BLOCK =
MLI: BE
ERROR? >>C00
GET MEM
IF ZERO 2 (VOLUME DIRECTORY) (2208)
ADD TWO AD BLOCK <BF05>
AND STOP >>21C3
ELSE, AT BLOCK NUMBER
WHEN 04, END OF VOLUME DIRECTORY >>21C8
PAGES (ONE BLOCK) TO POINTER
P AT $1400 IN ANY CASE
NEXT BLOCK AS WELL >>2197
XJ, JUMP SYSTEM FILE LOADER >>0800
226C $10/11 --> VOLUME DIRECTORY SY
226E INITIALLY AT $0800
2270 OFFSET BEYOND LINKS (+4)
2272 (TURN NEXT INSTRUCTION INTO 8

2273 *********** SYSTEM FILE LOADER ***********
2276 NEXT OBJECT ADDR: 2261
2279 (COPIED TO AND RUN AT $6000)

227C 227D 227E 227F
2280 2281 2282
2283 2284 2285
2286 2287 2288
2289 2290 2291
2292 2293 2294
2295 2296 2297
2298 2299 229A
229B 229C 229D
229E 229F 2300
2301 2302 2303
2304 2305 2306
2307 2308 2309
230A 230B 230C
230D 230E 230F
2310 2311 2312
2313 2314 2315
2316 2317 2318

22BB ---
22BD ---
22BE COPY NAME TO $281
22C5 AND TO "UNABLE TO LOAD" MSG (093B)
22CD ADD BLANK AT END OF NAME
22CF CHECK IN MESSAGE (093C)
22D3 NAMELEN + ERRORMSGLEN
22D5 SAVE AT $23BB (094F)
22DB MLI: OPEN .SYSTEM FILE <BF00>
22DC (PARM LIST AT $24BC)
22DE ERROR? >2326
22E0 MLI: GETEOF <BF00>
22E4 (PARM LIST AT $23C2)
22E6 ERROR? >2326
22E8 GET MSB (SEE $23C6) (095A)
22EB BIGGER THAN 64K???? >2340
22F0 MUST BE LESS THAN $9800 BYTES
22F2 OR ERROR... >2340
22F4 STORE LENGTH IN MLI READ LIST (0960)
22FA AND LFB TOO (095F)
22FD MLI: READ SYSTEM FILE INTO $2000 <BF00>
2301 (PARM LIST AT $23C7)
2303 NO ERRORS? >230B
2305 ERROR, BAD BUFFER?
2307 YES, FILE WAS TOO LARGE >2340
2309 ELSE, "UNABLE TO LOAD..." >2326
230B MLI: CLOSE SYSTEM FILE <BF00>
230F (PARM LIST AT $23CF)
230F ERROR? >2326
2313 NO, ENABLE MOTHERBOARD ROMS (C082)
2316 AND JUMP TO BEGINNING OF FILE >2300

2319 *********** ERROR HANDLERS ***********

2319 ---
231B PRINT "UNABLE TO FIND A .SYSTEM FILE" (08E2)
2324 THEN GO TO SLEEP >234B

2326 GET NAME LENGTH (094F)
2329 LINE LENGTH
232C LESS NAME LENGTH (094F)
232F DIVIDED BY 2
2330 GIVES OFFSET TO CENTER THE LINE (094F)
2334 PRINT "UNABLE TO LOAD..." (092A)
233E GO TO SLEEP FOREVER >234B

Beneath Apple ProDOS Supplement
Beneath Apple ProDOS Supplement

ProDOS Relocator -- V1.1.1 -- 19 SEP 84

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>2340</td>
<td>*** END ***</td>
</tr>
<tr>
<td>2342</td>
<td>PRINT &quot;SYSTEM PROGRAM TOO LARGE&quot; (890A)</td>
</tr>
<tr>
<td>234B</td>
<td>GO TO SLEEP FOREVER &gt;&gt;234B</td>
</tr>
</tbody>
</table>

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

234E ** UNABLE TO FIND A ".SYSTEM" FILE **
2346 ** SYSTEM PROGRAM TOO LARGE **
2348 ** UNABLE TO LOAD X.SYSTEM **********

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=0, CLOSE ALL FILES

23D1 '.SYSTEM'

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

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

(INCLUDES A ROUTINE AT $3D0 THAT COPIES CRITICAL ZERO PAGE VALUES TO AUX MEM)

23DB FROM MAIN Z-PAGE, (C000)
23DB GET X+1 VALUES STARTING AT $42
23E9 AT SAME LOCATION.
23E5 "NO DEVICE CONNECTED" ERROR
23EB BACK TO MAIN Z-PAGE (C000)
23E6 RETURN
23EC ADDRESS OF MLI ROUTINE
23F2 BRK HANDLER AT $F8A59
23F4 RESET AT $FF59
23F6 POWER UP BYTE
23F7 & VECTOR TO $FF59 >>FF59

ProDOS Reloc...

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>23FA CTProbe -- V1.1.1 -- 19 SEP 84</td>
<td>NEXT OBJECT ADDR: 23FA</td>
</tr>
<tr>
<td>23FD NES**********</td>
<td></td>
</tr>
<tr>
<td>2400</td>
<td>Incription/Contents</td>
</tr>
</tbody>
</table>

2402 ******

2402 VECTOR TO $FF59 >>FF59
2402 VECTOR TO $FF59 >>FF59
2402 HANDLER AT $BF00 (PRODOS)

2402 *****

2402 DETERMINE MACHINE ID ******

2402 $8000.. 0... APPLE II
2402 01.. 0... APPLE II+1
2402 10.. 0... APPLE Ile
2402 10.. 1... APPLE IIC
2402 11.. 0... APPLE /// EMULAT.
2402 11.. 1... 48K RAM
2402 ...10 .... 64K RAM
2402 111 .... 128K RAM
2409 AP$.... 1... 68 COL CARD
240B YES:.... 1... THUNDER CLOCK
240D NO:
240F AP$OME NOTHING AT FIRST
2411 YEA A ROM BYTE (FB03)
2413 NO$ IS IT?
2415 AP$, SET BIT >>242E
2417 NO:
241C RE$E IIE?
241E YEA; SET BIT >>242E
2422 //;
2426 --=- IIE?
2427 RE$ WHAT IS IT? >>242B
2428 TOTALLY A II+?
242A CR$ >>242E
242C AN$ EMULATION MODE?
242E UP?
2433 RETURN
2438 SELARWISE, UNKNOWN MACHINE
2444A IMA$E INVAL INSTR AT $80
244E G0 THERE >>244E

2451 ******MACHID

2451 WRITE ENABLE HIGH RAM (BANK1) (C000)
2451 IF HIGH RAM EXISTS (D00)
2451 PRESENT, MARK IN MACHID

2451 UR:
2453 I I******* LOOK FOR 64K OF AUX RAM ***********
2455 YR(CODE MOVED TO $80 TO ALLOW BANK SWITCH)
2457 BA(ENTERED WITH MACHID IN ACCUMULATOR)
245D ST
2460 AN$AYS MACHID
2466 MP$7, >>246A
2468 ITS:
2468 TO AUX MEMORY (C085)
2468 COM A PATTERN AT $C00 ($C00)
2468 TO $800 ($000)
2468 THE SCRE PATTERN STAYS THERE
2468 DIDN'T >>2478
Beneath Apple ProDOS Supplement

ProDOS Relocator -- V1.1.1 -- 18 SEP 84  
NEXT OBJECT ADDR: 246A

ADDR  DESCRIPTION/CONTENTS

246A  NOW SHIFT $C000 TO THE LEFT (0C00)
246D  AND SHIFT THE ACCUM TO THE LEFT
246E  ARE THEY STILL THE SAME? (0C00)
2470  NO, AUX RAM NOT THERE. >>2478
2473  DID $800 MOVE TOO? (0800)
2476  NO, SO WE HAVE FULL 128K1 >>247B
2478  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
249B  RETURN TO CALLER

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

2499  CLICK SPEAKER (C030)
249C  STORE IN MAIN MEMORY (C080)
249F  80 COL DISPLAY OFF (C000)
24A2  SET NORMAL VIDEO <FB84>
24A5  CALL MONITOR INITIALIZATION <FB2F>
24AB  SET VIDEO PR0 & <FE93>
24AE  OUT OF DECIMAL MODE
24AF  DISABLE FOR INTERRUPTS
24B0  CLEAR SCREEN <FC5B>
24B5  PRINT "APPLE /" (24E3)
24C0  PRINT "PRODOS 1.1.1 ETC." (24EB)
24CB  PRINT A BLANK AT $6AB (2502)
24DD  PRINT "COPYRIGHT ETC." (2503)
24DF  CLICK SPEAKER AGAIN (C030)
24E2  DONE

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

24E3  'APPLE /'
24E5  'PRODOS 1.1.1 18-SEP-84'
2502  '
2503  'COPYRIGHT APPLE COMPUTER, INC., 1983-84'

252A  ********** DETERMINE SLOT CONFIGURATION **********

252A  ---
252C  ZERO SOME THINGS
2533  NO DISKS ACTIVE YET (BF31)
253B  $A/0/11 --> $C700 (LOOP THRU ALL SLOTS)
253A  RESET I/O CARD ROMS (C0FP)

253F  CHECK SIGNATURE ON CARD FOR DISK DEVICE
2545  DISK? >>25AD
255F  $BBF BYTE [TYPE OF DISK]
2567  $117 >>256F
256B  $B PROFILE?
256D  THEN NOT A DISK >>25AD
255F  NO

2551  ********** PROFILE FOUND **********

2555  E, SAVE AS LSB OF BLOCK READ SUBRNT
255C  $BBF (STATUS BYTE)
255E  WE AT LEAST READ STATUS AND DATA?
2563  >>2563
2559  A DISK AFTER ALL >>25AD
2562  STATUS BYTE AGAIN
256E  NIBBLE IS DEVICE ID
2561  FILE SHOULD BE $04
2567  COUNT NUMBER OF VOLS (SHOULD BE 0)
256B  FE SLOT NO. FOR DEVICE DRIVER LOC.
256A  GO DO COMMON PROCESSING FOR DISK >>2579
256B  GET

256D  AMP

256F  ZERO FOR DISK II

2571  DISK II DEVICE DRIVER LOCATION (266A)
2571  $200 OR $2DB0 (266B)
2571  GET

2575  DISK FOUND **********

2579  $A6 LD ST (S=SLOT, $D=DISK1,4 PROFILE)
257B  $F=DEVICE COUNT BY ONE (BF31)
2576  NUMBER OF DRIVES
2586  AM I ONE? >>2596
258A  BUMP INDEX
2595  MARK SECOND DRIVE IN SEARCH LIST (BF32)
2590  MARK SECOND DRIVE IN SEARCH LIST (BF32)
2592  RE FINAL DEVICE COUNT (BFJ1)
2593  UP DISK DEVICE DRIVER VECTORS (BF11)
2596  SYSTEM GLOBAL PAGE >>25AB
2597  SET UP TWO VECTORS FOR A DISK II (BF21)
2597  SET
2598  IN

25A0  RECOGNIZE THIS CARD
25A8  MARK SLTBYTE TO SHOW ROMS IN SLOT <25F>
25AC  ALL CARDS EXCEPT
25AD  T & ($C000) >>253A

25A4  LAST DISK DEVICE IN SEARCH LIST (BF32)
25B6  $D DRIVE? (BF30)
25BC  GEF
25C2  B0C
Beneath Apple ProDOS Supplement

ProDOS Relocator — V1.1.1 — 18 SEP 84

<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>25CA</td>
<td>---</td>
</tr>
<tr>
<td>25CD</td>
<td>GET DEVICE COUNT (BF31)</td>
</tr>
<tr>
<td>25E1</td>
<td>IS BOOT DRIVE IN LIST? &gt;&gt;25E7</td>
</tr>
<tr>
<td>25E2</td>
<td>SO IT WILL BE SEARCHED FIRST... (BF'36)</td>
</tr>
<tr>
<td>25E5</td>
<td>STORE BOOT AT END OF SEARCH LIST (BF32)</td>
</tr>
<tr>
<td>25DA</td>
<td>ANY OTHERS? &gt;&gt;25EE</td>
</tr>
<tr>
<td>25DD</td>
<td>YES, SECOND DRIVE? &gt;&gt;25E7</td>
</tr>
<tr>
<td>25E8</td>
<td>STORE IT RIGHT BEHIND BOOT DRIVE (BF32)</td>
</tr>
<tr>
<td>25E7</td>
<td>NOW ANY MORE? &gt;&gt;25EE</td>
</tr>
<tr>
<td>25E7</td>
<td>---</td>
</tr>
<tr>
<td>25E9</td>
<td>YES, MOVE OTHERS AHEAD IN LIST (BF32)</td>
</tr>
<tr>
<td>25E9</td>
<td>DO CHECKSUM ON ROM &lt;267C&gt;</td>
</tr>
<tr>
<td>25F1</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>25F5</td>
<td>AND LEAVE</td>
</tr>
<tr>
<td>25F7</td>
<td>NOT AUTOSTART, UNKNOWN MACHINE, SO CRASH! &gt;&gt;242B</td>
</tr>
</tbody>
</table>

25FA ************** IDENTIFY I/O CARD **************

25FA | DO WE ALREADY RECOGNIZE THIS CARD? >>265B |
25FC | NO, |
25FE | CHECK SIGNATURE ON CARD FOR THUNDER CLOCK |
2603 | NOT IT? >>261F |
2609 | THUNDER CLOCK, WHICH SLOT? |
261B | SAVE SLOT NUMBER (LESS 1) |
261C | IN CLOCK CODE RELOCATION TABLE (226A) |
2612 | ENABLE CLOCK/CALENDAR JUMP IN GLOBALS (BF06) |
2617 | IS THERE A MACHID? >>25EE |
2619 | IF SO, MARK THAT A CLOCK IS PRESENT |
261B | AND UPDATE MACHID |
261D | Condition ROM IN THIS SLOT >>265B |
261F | --- |
2621 | CHECK SIGNATURE OF MYSTERY CARD |
2623 | STANDARD BASIC SUPPORTED? |
2625 | NO, UNKNOWN CARD >>264A |
2629 | YES, |
262B | DOUBLE CHECK BASIC SUPPORTED |
262D | NO, UNKNOWN CARD >>264A |
2631 | YES, |
2633 | GENERIC SIGNATURE? |
2635 | NO, UNKNOWN CARD >>264A |
263B | YES, |
263C | 80 COLUMN CARD? |
263E | NO, UNKNOWN CARD >>264A |
2642 | GET MACHID IF WE HAVE ONE >>25EE |
2644 | MARK 80 COLUMN CARD PRESENT |
2646 | AND UPDATE MACHID |
264B | GO MARK ROM ON CARD PRESENT >>265B |
264A | UNKNOWN CARD, CHECK ROM TO... |
264E | SEE IF IT WILL HOLD A VALUE... |

264A ************** RELLOCATION (X/Y REGS CONT) |

2654 | FOR SOME TIME. |
2655 | IF SO, WE HAVE A CARD |
265D | CONVERSE SLOT NUMBER. |
2660 | TO A BIT POSITION (2 |
2663 | AND OR INTO SLTBYZ ( |
2669 | RETURN TO CALLER |
2677 | (ALSO USED IN CHECK |
2674 | BIT POSITION TABLE F >> INPUT BLOCK |
2677 | ) |

267C ************** COMPUTE AUT...

267C | --- |
267D | GET ZERO IN INDEX REG |
2686 | SUM $8B89 ("APPLE II") |
2687 | UPDATE CHECKSUM (267C) |
268E | DO 8 BYTES IN ALL (X) |
2694 | MOVE LENGTH TO HIGH |
2699 | AND COMBINE WITH UCHAR |
269C | FUDGE FACTOR |
269E | SHOULD COME OUT ZERO... |
26A0 | IT DID... RETURN WITH |
26A2 |ὴ RETURN |
26A3 | ELSE, RETURN WITH Ŕ | 17 |
26A5 | RETURN |

26A6 ************** RELATION (X/Y REGS CONT) |

26A6 | SAVE PASSED TABLE ADDR >>18 SEP 84 |
26A6 | NEXT OBJECT ADDR: 2654 |
26A8 | --- |
26AC | GET OPERATION CODE |
26AE | VALID Operation? (4 |
26B0 | NO, ERROR >>2724 |
26B4 | $14/15 --OUTPUT BD IN SLOT |
26BE | $16/17 -- LENGTH |
26C7 | NEGATIVE LENGTH? >>74 |
26CA | CHECK OPERATION CODE(399) |
26CD | ZERO BLOCK? >>272F |
26CD | NO, $12/13 = $18/19 |
26D7 | $1A/18 -- END OF 10 |
272F *********** 0 - ZERO BLOCK *************************
2732 BUMP TABLE POINTER TO NEXT ENTRY <2759>
2734 GET NUMBER OF PAGES TO DO
2736 NO FULL PAGES? >>2744
2739 ZERO AN ENTIRE PAGE
273E BUMP PAGE POINTER
2740 AND DECREMENT LENGTH
2744 GET LENGTH OF PARTIAL LAST PAGE
2746 NO PARTIAL PAGE? >>2750
2749 ZERO PARTIAL PAGE TOO
2750 DONE, GET NEXT TABLE ENTRY >>26AA

2753 *********** 1 - COPY BLOCK *************************
2753 BUMP TABLE POINTER <2759>
2756 AND GO COPY BLOCK >>271E

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

Beneath Apple ProDOS Suppilment

ProDOS Relocator -- V1.1.1 -- 18 SEP 84 NEXT OBJECT ADDR: 26E4

26B4 COPY BLOCK ONLY? >>2753
26B6 SAVE RELLOCATION OPERATION CODE (287F)
26B8 SAVE NUMBER OF RANGES TO CHECK (2880)
26F0 ---
26F1 COPY START PAGES TO TABLE
26F2 ---
26F4 AND END PAGES
2708 ---
2709 AND FINALLY, RELLOCATION FACTORS
2711 BUMP TO NEXT TABLE ENTRY <2759>
2714 RESTORE OPERATION CODE (287F)
2719 RELOCATE INSTRUCTIONS? >>2729

271B *********** 2/3 - RELOCATE ADDRESSES **********************
271B NO, RELOCATE ADDRESS <27BD>
271E COPY BLOCK <2766>
2721 AND CONTINUE IF ALL WENT WELL >>26AA
2724 NORMAL EXIT
2725 RETURN
2726 JUMP TO ERROR EXIT >>27F3

2729 *********** 4 - RELOCATE INSTRUCTIONS **********************
2729 RELOCATE INSTRUCTIONS <27CF>
272B ADD FINAL ENTRY INDEX...
272D TO TABLE ENTRY ADDRESS
2730 RETURN

2730 *********** COPY BLOCK *************************
2766 ---
276A INPTR < OUTPTR? >>2777
276C NO, GREATER? >>279A
276E MSB'S ARE EQUAL, CHECK LSB'S ALSO
2776 EXIT IF EQUAL
2777 INPTR < OUTPTR, COPY LAST PAGES FIRST
277B BUMP BOTH INPTR AND OUTPTR BY...
277D LENGTH-1 TO POINT AT LAST BYTE
2781 START WITH SHORT LAST PAGE LENGTH
2788 ---
278A COPY BYTES BACKWARDS THROUGH MEMORY
2791 DROP ADDRESSES AND LENGTH BY 256
2797 AND CONTINUE UNTIL FINISHED >>2789
2799 RETURN
279A INPTR > OUTPTR, COPY PAGES FORWARD
279C HOW MANY FULL PAGES LEFT?
279E NONE? >>27AF
27A0 COPY A FULL PAGE
27A7 AND BUMP ADDRESSES
27B0 DECREMENT LENGTH BY 256
27B2 AND DO ALL PAGES >>27AB
27AF GET LENGTH OF LAST PAGE
27B1 EVEN PAGE BOUNDARY? >>27BC
27B3 NO, COPY SHORT LAST PAGE
27BC RETURN

27BD GET TABLE ENTRY TYPE (287F)
27C1 GET PAGE TO RELOCATE
27C3 RELOCATE A SINGLE ADDRESS <27FB>
27CF BUMP BY 1 OR 2 BYTES (287F)
27C9 ADVANCE POINTER <2817>
27CC AND CONTINUE UNTIL COMPLETE >>27BD
27CE RETURN

27CF ---
27D1 GET 6502 OPCODE
27D3 COMPUTE INSTRUCTION LENGTH <282A>
27D6 INVALID OPCODE? >>27E9
27D8 3 BYTE INSTRUCTIONS?
**COMPUTE INSTRUCTION LENGTH**

**CONTAINS OPCODE**

**LAST TWO BITS FOR LATER**

**TOP TWO BITS AS INDEX**...**>>283C**

**RELOCATE IT IN A-REG**

**IS IT PRIOR TO START**

**GET NUMBER OF RANGE:**

**IS IT AFTER EN#**

**GET PUDGE FACTOR TO**

**ADD LENGTH TO POINT**: CHECK TO SEE IF WE!

**NEXT OBJECT ADDR: 27DA**

**3 BYTE ADDRESS**

**RELOCATE ADDRESS <<2**

**CONTINUE UNTIL FINI**

**RETURN TO CORRECT**

**ADD ADVANCE BY 3 BY**

**NEXT INSTRUCTION <<2**

**RETURN WITH BAD INSTRUC.**

**RETURN WITH ERROR CODE**

**RELOCATE AD**

**RETURN**

**DISTINCT ADDRESS**

**LESS ONE**

**ADD PUDGE FACTOR TO**

**ADJUST ALERT**

**RETURN**

**BUMP POIN**

**ADDRESS (2891)**

**CHECK TO SEE IF WE!**

**TERK TO NEXT ADDR**

**RETURN**

**ARE DONE**

**RETURN**

**TOP TWO BITS AS INDEX**

**USE 4 LENGTHS IN IT (283F)**

**GET B**

**USING**

**SHIFTP3**

**RELOCATION DATA**

**OF RANGES**

**EACH RANGE PAGES +1**

**FIVE FACTORS**

**G499-28FE NOT USED**

**RELOCATE IN RAMDRIVE IN AUXMEM**

**STARTS ROUTINE PUTS THE RAMDRIVE DEVICE DRIVER**

**END OF MEMORY, PUTS THE ADDRESS OF THE DRIVER**

**ADD THE DEVICE DRIVER ADDRESS LIST, AND**

**IS THE RAMDRIVE TO THE ONLINE DEVICE LIST.**

**NOT A CALL**

**CH RAM AT. (FF08)**

**SUB MAIN MEM TO AUX MEM**

**SUB MOVE TO COPY IT (C311)**

**TO HIDE DRIVE 2 DEVICE DRIVER.. (BP26)**

**NOW $FF08**

**DEVICE COUNT (BF31)**

**INTO DEVICE TO ONLINE DEVICE LIST**

**$1C/0$**

**$3E/0$**

**$42/0$**

**$4C$**

**USE**

**SLOT**

**IS A**

**BUMP**

**ADD**

**RETUR**
2A00 ****** RAMDRIVE (/RAM) DEVICE DRIVER ***************

(COPIED TO AND RUN AT $200 IN AUX RAM)
(this is the main part of the device driver.
it is called by the ramdrive caller
which is located at $F000 in main memory.)

2A00 SAVE THE 8STORE SETTING (C018)
2A04 FORCE RAM READ/WRITE (C080)
2A09 COPY INPUT PARAMETERS
2A0B TO AUX PAGE 3. (03BD)
2A11 FIRST TIME IN OR FORMAT COMMAND? (03BC)
2A14 NO, SKIP FORMAT LOGIC >>2A4F

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

2A16 YES, SAVE BLOCK WANTED
2A18 PAGES $E AND $F ARE ACTUAL DIRECTORY
2A1A ZERO THE DIRECTORY BLOCK (0333)
2A1F COPY VOLUME NAME ($F3, "RAM") (03D2)
2A22 TO VOLUME DIRECTORY BLOCK (0E04)
2A28 LAST BYTE IN VOLUME_BITMAP
2A2A IS AN $FE (03D1)
2A2D $FF TO ACCUM.
2A30 14 $FF'S TO BITMAP (03C2)
2A36 SET FIRST BITMAP BYTE TO ZERO (03C2)
2A39 COPY 8 BYTES
2A3B OF DIRECTORY DATA (03D6)
2A3E TO VOLUME DIRECTORY BLOCK (0E22)
2A44 WAS THIS A FORMAT COMMAND? (03BC)
2A47 YES, DONE. >>2AA
2A49 NO, SET FLAG & CONTINUE WITH READ/WRITE (03BC)
2A4C RESTORE BLOCK NUMBER (03C1)

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

2A4F CONVERT BLOCK NUMBER TO PAGE NUMBER (03C1)
2A55 THIS PAGE IN HIGH RAM?
2A57 YES >>2A63
2A59 NO, IS IT BLOCK 37 (VOLUME BIT MAP)
2A5B NO >>2A60
2A5D YES, DUMMY UP A PHONY BITMAP BLOCK >>038C
2A60 ELSE, NORMAL READ/WRITE >>0342
2AE5 ************ SET BUFFER AND BLOCK ADDRESSES ******************
2AE5 GET COMMAND (03BD)
2AE8 READ OR WRITE?
2AE9 WRITE? >>2B08
2AE8 NO, GET HIGH BYTE OF BUFFER TO BE READ (03Cu)
2AFA AND LOW BYTE OF BUFF ADDRESS (03BF)
2AF5 $42/43 --> FIRST PAGE OF BUFFER
2AF7 $40/41 --> SECOND PAGE OF BUFFER
2AF9 GET PAGE NUMBER (03C1)
2AFC $3C/3D --> BLOCK IN RAMDRIVE
2B08 $3E/3F --> SECOND PAGE OF SAME
2B0E ALWAYS BRANCH AROUND WRITE CODE >>2B23
2B08 WRITE, (03C0)
2B0F $3C/3D --> MAIN MEMORY ADDRESS OF BUFFER TO BE WRITTEN (03BF)
2B12 $3E/3F --> SECOND PAGE OF SAME
2B19 $42/43 --> BLOCK IN RAMDRIVE
2B1B $40/41 --> SECOND PAGE OF SAME
2B23 SET SECOND PAGE ADDRESSES
2B27 EXIT

2B2B *********** SEND HIM A DUMMY BLOCK OF ZEROES*******************
2B2B ZERO RAMDRIVE BUFFER IN CASE READING <0331>
2B2B COPY BETWEEN RAMDRIVE BUFFER AND HIS BUFFER <02C3>
2B2B AND EXIT >> 03DE

2B31 *********** ZERO BLOCK BUFFER *******************************
2B31 ZERO RAMDRIVE BUFFER
2B33 ZERO BLOCK INDICATED BY ACCUM. (03C1)
2B36 SET UP BUFFER POINTERS <02E5>
2B3A ZERO BOTH PAGES OF BLOCK
2B41 AND EXIT

2B42 ************ READ/WRITE IN LOW 48K **********************
2B42 BLOCK 2 (VOLUME DIRECTORY)?
2B44 NO >> 2B4A
2B46 YES, CONVERT IT BLOCK 7
2B4B AND GO DO I/O NOW >> 2B5B
2B4A ELSE, LESS THAN BLOCK 87 (BUG--$D SHOULD BE $F111)
2B4C YES, RETURN WITH DUMMY ZERO BLOCK. >> 2B28
2B4E START MSB AT ZERO
2B50 GET ORIGINAL BLOCK NUMBER
2B52 BLOCK $50 THROUGH $5F?
2B54 NO >> 2B5B
2B56 YES, ADJUST TO $0 THROUGH $F
2B58 AND USE $1A00 THRU $1FFF IN RAMDRIVE. >> 0385
ProDOS Relocator -- v1.1.1 -- 18 SEP 84

ADRR DESCRIPTION/CONTENTS
2C44 NORMAL EXIT, RETURN CODE IS 0
2C47 ---
2C4B RESTORE ZERO PAGE (FF61)
2C53 AND $3ED/E (FF7F)
2C61 AND EXIT TO CALLER WHEN THRU
2C62 *********** COPY MAIN TO AUX BLOCK ***********
(CALLED FROM AUX MEM HANDLER)

FF62
2C62 WRITE IN AUX 48K (C005)
2C67 COPY BOTH PAGES OF BLOCK
2C72 WRITE IN MAIN 48K AGAIN (C064)
2C77 GO TO $20A IN AUX MEMORY TO RETURN ($0ED)
2C7C RETURN TO AUX MEM HANDLER AGAIN >>FF33
2C7F *********** DATA AREA ***********

FF7F
2C7F SAVE $3ED,$3EE
FF80
FF81
2CG1 ZERO PAGE SAVE AREA

2C8D *********** $2C8D-$2CFF NOT USED ***********
2C8D NOT USED
2CA0

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

ProDOS Relocator -- v1.1.1 -- 18 SEP 84

ADRR DESCRIPTION/CONTENTS
2C44 NORMAL EXIT, RETURN CODE IS 0
2C47 ---
2C4B RESTORE ZERO PAGE (FF61)
2C53 AND $3ED/E (FF7F)
2C61 AND EXIT TO CALLER WHEN THRU
2C62 *********** COPY MAIN TO AUX BLOCK ***********
(CALLED FROM AUX MEM HANDLER)

FF62
2C62 WRITE IN AUX 48K (C005)
2C67 COPY BOTH PAGES OF BLOCK
2C72 WRITE IN MAIN 48K AGAIN (C064)
2C77 GO TO $20A IN AUX MEMORY TO RETURN ($0ED)
2C7C RETURN TO AUX MEM HANDLER AGAIN >>FF33
2C7F *********** DATA AREA ***********

FF7F
2C7F SAVE $3ED,$3EE
FF80
FF81
2CG1 ZERO PAGE SAVE AREA

2C8D *********** $2C8D-$2CFF NOT USED ***********
2C8D NOT USED
2CA0

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

ProDOS Relocator -- v1.1.1 -- 18 SEP 84

ADRR DESCRIPTION/CONTENTS
2C44 NORMAL EXIT, RETURN CODE IS 0
2C47 ---
2C4B RESTORE ZERO PAGE (FF61)
2C53 AND $3ED/E (FF7F)
2C61 AND EXIT TO CALLER WHEN THRU
2C62 *********** COPY MAIN TO AUX BLOCK ***********
(CALLED FROM AUX MEM HANDLER)

FF62
2C62 WRITE IN AUX 48K (C005)
2C67 COPY BOTH PAGES OF BLOCK
2C72 WRITE IN MAIN 48K AGAIN (C064)
2C77 GO TO $20A IN AUX MEMORY TO RETURN ($0ED)
2C7C RETURN TO AUX MEM HANDLER AGAIN >>FF33
2C7F *********** DATA AREA ***********

FF7F
2C7F SAVE $3ED,$3EE
FF80
FF81
2CG1 ZERO PAGE SAVE AREA

2C8D *********** $2C8D-$2CFF NOT USED ***********
2C8D NOT USED
2CA0

2D00 *********** START OF PRODOS LOAD IMAGE ***********
2D00 LOAD IMAGE AT $2D00
ProDOS MLI -- V1.1.1 -- 18 SEP 84

NEXT OBJECT ADDR: D700

D700 MODULE STARTING ADDRESS

******************************************************************************
* *
* PRODOS MACHINE LANGUAGE INTERPRETER
* THIS CODE IS MOVED INTO RAM ($D800-$F7EF) BY THE PRODOS RELOCATOR.
* IT PERFORMS ALL FILE MANAGEMENT AND OTHER SYSTEM FUNCTION
* SUPPORTS THE HARDWARE IN DEVICE INDEPENDENT WAY.
* VERSION 1.1.1 -- 18 SEP 84
*
******************************************************************************

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

0040 Pointer to callers parm list
0041 -- device driver parm list --
0042 Command
0043 Unit Number
0044 Buffer Pointer
0045
0046 Block Number
0047
0048 I/O Pointer - Index Block or...
0049 caller's pathname buffer pointer
004A I/O Pointer - Data Block
004B I/O Pointer - Data Block
004C I/O Pointer - Data Block
004D I/O Pointer - Caller's Data or...
004E buffer pointer passed in parm list
004F or...

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

0000 No Error
0001 Bad call type
0004 Bad parameter count
0025 Interrupt Table full
0027 I/O Error
0028 No device connected
002B Write protected

For direct movement...

0115 version

Slot in use

********** SYSTEM GLO****

Jump to MLI entry...
JSPARE (Jump to $E883)
DATETIME vector
Jump to System Err...
Jump to System Dea...
System Error number
Device Driver addr
Slot/Drive last de...
Count (-1) active ACTIONS
List of active dev...
Memory BITMAP for text to screen
Open file 1 buffer

********** REAL PAGE EQUATES **********

0750 MLI -- V1.1.1 -- 18 SEP 84

NEXT OBJECT ADDR: D700

07F2
07F3 Volume switched
07F4 Invalid pathname sy...
07F5 Too many files open
07F6 Invalid REP NUM
07F7 Nonexistent path
07F8 Volume not mounted
0800 Duplicate file name...
0801 Disk full
0802 Volume Directory full
0803 Incompatible ProDOS
0806 Unsupported file foo...
0809 End of file
080C Position past EOF
080F Access error
0810 File already open
0813 File count bad
0815 Not a ProDOS disk
0831 Bad parameter
0832 VCB overflow
0835 Bad buffer address
083B Duplicate volume
083C Bad volume bit map

********** SCREEN LOCATION

23
buffer address
dier 1
dier 2
dier 3
dier 4
ring interrupt
ring interrupt
ring interrupt
ring interrupt
urn address
EL
Ø = no prefix)
tag
return address
area
area
exit routines
exit routines
saved state ($E000 byte)

SWITCHES ****************************************
mm mode
/t/graphics
ty page
physic mode
to I/O ROMs

AME - DATA AREA ******************************
----

it top of buffer such that a
dex may be used to use it,
ound to the pathname again.
er

CONTROL BLOCKS ******************************

here.
er

THE FOLLOWING 6 BYTES ARE THE FILE ID

D801 Device Number
D802 Dir Block HDR for Dir describing this File
D804 Dir Block containing entry itself
D806 File entry # in this Directory

D807 Storage Type

Flags
LXXX XXXX Index Block Buffer Changed
X1XX XXXX Data Block Buffer Changed
XX1X XXXX Unused
XXX1 XXXX Directory entry needs update
XXXX XXXX Storage Type Changed
XXXX XXXX Allocate new Master Index Block
XXXX XXIX Allocate new Sub index Block

D808 XXXX XXXX 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
D816 Blocks Used
D81A not used
D81B Level
D81C Flag - Write occurred if MSB on
D81D not used
D81F Newline Enable Mask
D820 FCB1 through FCB7

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

VCB0 starts here...

D900 Length (0000H)
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

Block offset into multi-block bitmap of
next free bit.

D91C Count of open files
ProDOS MLI -- V1.1.1 -- 18 SEP 84

ADDR DESCRIPTOR

D920 VCB1 t 2nd half
DA00 FIRST PRIMARY BUFFER ****************************
D800 Buffer fields
DC00 VOLUME DIRECTORY HEADER ***

DC00 Pointer: (TTTTTTTTT)
DC04 Type/Length
DC08 Volume Length
DC12 Reserved
DC14 Creation
DC18 Creation
DC20 Vers
DC21 Min Vblocks
DC22 Access (offset of first page of block)
DC24 Entry (page of block)
DC2C Entry (main entry point)
DC30 File C
DC34 File A
DC38 File D
DC44 File T
DC48 File Y
DC52 Exit to caller >>DE71

DE54 Else, $4X - Interrupt support
DE55 Isolate type (DEALLOC = 1, ALLOC = 0)
DE57 Call Interrupt Support <DEF3>
DE5A Then Exit to Caller >>DE78
DE5D Go to quit code via global page >>BF00

DE60 MLI GET TIME CALL ******

DE66 MLI READ BLOCK CALL ******
DE67 MLI WRITE BLOCK CALL ******

$80 - Read Block
$81 - Write Block

DE6D Set $42 -> 1 for READ, 2 for WRITE
DE6F Do Block I/O <DEE2>
DE70 Then Exit to Caller >>DE78

DE71 $CX and $DX CALLS ******

DE75 Perform function and exit to caller <E047>

DE78 EXIT TO CALLER ******

DE80 Clear Backup
DE8D Error occurred?
DE93 Save test results
DE94 Disable interrupts
DE95 MLI no longer active (BF98)
DE9B Test results back
DE9D Store in X reg
DE9A Set up Return Address on stack (BF9D)
DE9F Put test results on stack
DE94 Put error code in A reg
DE9F Restore X reg (BF9E)
DE9F Restore Y reg (BF9F)
DE9B Put error code on stack
DE9C Get RAM/ROM orientation (BF4)
DE9F Exit via RAM Global Page >>BF00
DEA2 ******* NO DEVICE CONNECTED ****************************

DEA2  
DEA4 Call System Error Handler (Global Page) <BF99>

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

DEA7  
DEA9 Branch always taken >DEAD

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

DEA8  
DEA9 Call System Error Handler <DED7>

DEBU Exit to Caller >DEB8

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

DEB2  
DEB4 Save Old Processor Flags

DEB5 Disable Interrupts

DEB6 Copy Parameters to $43-$47

DEB7 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>

DECC Yes, then exit with error >DED6

DECE No, do Block I/O <DEDA>

DED1 Error? >DEDA

DED3 No, then exit normally

DED5 RETURN

DED6 Error Exit

DED7 Call System Error Handler <BF99>

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

DEDA  
DEDC Force off enaged UNIT bits

DEE7 Put Device Handler Address in Jump Vector (FEF5)

DEF0 Exit through Device Handler >FEF5

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

ALLOC/DEALLOC

DEF3 Save Call Type

DEF5 Which Type?

DEF6 DEALLOC? >DEF4

DEF24 Get Position Number

DEF8 Can't be zero >DEF1F

DEF9 Or greater than 4 >DEF1F

DEFB Make Index into Table from it

DEFD And zero his Vector (BF7E)

DEFA Skip this Vector

DEFA Least one?

DEFB No, check another >DEFA

DEFB Yes, Table Full Error

DEFG Always taken >DF21

DEFI Bad Parameter Error

DF21 Call System Error Handler <BF99>

DEFA ******* IRQ Handler ****************************

DF3A  
DF3C Save A reg from Monitor (BF88)

DF3F And $X, $Y and P (BF89)

DF49 Is this ROM enhanced? (DF8)

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 >DF91

DF7C Yes, call it <DF9D>

DF7F His interrupt? >DF4A

DF81 Is there a User Vector #2 (BF83)

DF84 No >DF8B

DF86 Yes, call it <DFDC>

DF89 His interrupt? >DF4A

DF8B Is there a User Vector #3 (BF85)
Beneath Apple ProDOS Supplement

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

---

DF8E No >> DF95
DF90 Yes, call it <<DFDF
DF93 His interrupt? >> DF4A
DF95 Is there a User Vector #4? (DE)
DF98 No >> DF9F
DF9A Yes, call it <<DFE2
DF9D His interrupt? >> DF4A
DF9F Indicate error type 1

DFA1 Call System Death Handler <<DFA4
DFA4 Interrupt Serviced
DFA6 Restore zero page (FDF5)
DFAE And stack (BF6B)
DFBE Is this enhanced ROM? (DFD8)
DFC1 Yes, skip some stuff we use
DFC3 Reload X and Y (BF6A)
DFC9 Disable I/O ROMS (CFFP)
DFCC Replace active slot number
DFD5 Exit from interrupt >> BFDO
DFD8 ENHANCE FLAG. Set to 1 by ROM

DFD9 User Interrupt Handlers (4)

**SYSTEM ERROR HANDLER**

DFE5 Save Error Code (BF07)
DFE9 Pop out of subroutine
DFEB Exit to caller with Error Code
DFEE RETURN

**SYSTEM DEATH HANDLER**

DFEF ---
DFF1 Entry from System Global Page
DFF2 Turn off 80 column card (C007)
DFF3 Select standard Text display
E001 Blank out a line
E003 ---
E006 Print "INSERT SYSTEM DISK AT" and "error code 00"
E012 go into infinite loop if no NAME
E016 "-" (07F1)
E01B "E" (07F2)
E020 "R" (07F3)
E021 "R" (07F4)
E027 Convert error code to Hex
E033 And print it (07F6)
E037 Second digit also
E044 Infinite loop >> E044

---

NEXT OBJECT ADDR: E044

---
E00A No - get next character in his name
E00B Is it "/"?
E00C Yes >>E14
E00D No - lower case?
E00E No >>E0DA
E00F Yes - force upper case
E010 Copy to my Pathname buffer (D703)
E011 Increment Index level counter (FEB8)
E012 Subsequent characters may be A-Z, 0-9 or . >>E0E7
E013 Increment Index level counter (FEB8)
E014 First character must be alphabetic >>E0F3
E015 Is it "."?
E016 Yes - get next character >>E0C5
E017 No - is it special or control character
E018 Yes - Bad Pathname then >>E0FB
E019 Is it numeric?
E01A Yes - get next character >>E0C5
E01B Is it Alphabetic?
E01C If so get next character >>E0C5
E01D Else
E01E Bad Pathname
E01F RETURN
E020 ----
E021 Any characters in last Index level? (FEB8)
E022 Yes >>E10A
E023 No, zero characters in it (FEB8)
E024 And toss out last "/"
E025 ----
E026 Mark end of name with $00 (D708)
E027 Name too long? >>E0FB
E028 No - save final length (FEBE)
E029 Set X -> 0
E02A Last Index more than 15 characters?
E02B Yes - then no good >>E0FB
E02C Save output Index (FEBD)
E02D Store length of previous Index level (FEBA)
E02E Just before it in buffer (D706)
E02F Restore output index (FEBD)
E030 And continue >>E0BA
E031 End of Name
E032 Fully qualified name? (FEBC)
E033 Yes >>E134
E034 No - Got a Prefix (BF9A)
E035 No - error >>E0FB
E036 Else, okay to exit
E037

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

E135 Copy Pathname <E08A>
E136 It's okay >>E144
E137 Check length of Volume name (D700)
E138 If zero - no Prefix wanted (BF9A)
E140 Exit with no error
E141 RETURN
E142 Get File entry for last Index <E5A3>
E143 Okay? >>E14D
E144 Invalid Pathname?
E145 No - Out now! >>E18B
E146 Sub Directory file? (FESF)
E147 No, error >>E169
E148 Fully qualified path? (FEBC)
E149 Yes >>E156
E150 No - use old Prefix also (BF9A)
E151 Compute new Prefix Index (FEB9)
E152 Does new Prefix exceed 64 characters?
E153 Yes - Bad Path error >>E0FB
E154 Store new Prefix pointer (BF9A) (FESF)
E155 Set Device Number of Prefix Directory (.)
E156 Save Keyblock for Prefix Directory (FEB4)
E157 Copy Prefix to top of Path buffer (D70E,D700)
E158 (preceded by old Prefix if one exists)
E159 Exit normally
E160 ----
E161 Bad File Type Error
E162 ----
E163 RETURN

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

E18D Set ($4E) -> Data Buffer
E199 Set Length = 64 (max)
E1A3 Validity check buffer storage <FCS2>
E1A6 Error? >>E18B
E1AA Get Prefix index (BF9A)
E1AE No Prefix? - Length = 0 >>E1B4
E1B0 Complement for length
E1B4 Store in first byte of buffer
E1B6 If null Prefix exit >>E1CE
E1BB ----
E1BC Copy Prefix to caller's buffer replacing
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>
</tbody>
</table>

**E1D0**  ********** VALIDITY CHECK REFERENCE NUMBER ************
(PASSED BY CALLER)

- E1D0 Get Reference Number
- E1D4 If zero then no good >>E231
- E1D8 If > 8 then no good >>E231
- E1DA Save Reference Number
- E1DB Multiply by 32
- E1E1 Result gives offset into PCB's (FE92)
- E1E5 Get back Reference Number
- E1E6 File Control Block active this Reference? (D800)
- E1E9 No -> Bad Reference Number >>E22C
- E1EB Get Buffer Number (D80B)
- E1EE Find Buffer address in Global Page <FC3C>
- E1F4 No Buffer? >>E21D
- E1F6 Buffer okay, save Page Pointer in $48
- E1FA Second block in $49
- E1FC Set last Device used in Global Page (D801)
- E202 Finish setting up pointers (FE8D)
- E205 ($4A) -> 1st Block of Buffer (data)
- E207 ($4B) -> 2nd Block of Buffer (index)
- E209 ---
- E20A Search all Volume Control Blocks (D910)
- E20D For the one which goes with requested unit (D801)
- E212 ---
- E218 Can't find matching Volume Control Block
- E21A So die with error type $0A <BF0C>
- E21D No Buffer in open File Control Block
- E21F So die with error type $0B <BF0C>
- E222 Is Volume mounted? (D900)
- E225 No, keep looking >>E212
- E227 Save Volume Control Block index (FE91)
- E22B Exit normally

**E22C ---**

**E22E This looks wrong!!! (FE92)**

**E231 Bad Reference Number error**

**E234 RETURN**

**E235 ****** MLI ONLINE CALL ******

* ****************************
E23C *** ERROR ***

E23C Store Code in Data Buffer Entry

E23C Store Device Number in Buffer Entry (E32C)
E23C Store error code next
E23C Duplicate Volume Entry
E23C No - done >> E23F
E23D Store Device Number in Buffer Entry (E32C)
E23B No Duplicate now
E23F Exit with error
ProDOS MLI -- V1.1.1 -- 18 SEP 84

ADDR DESCRIPTION/CONTENTS

E3B4 ********** ZERO $F600 ***********************************************
E3B4 Zero $F600 Block Buffer
E3C0 Return to caller

E3C1 ********** BUILD NEW FILE *********************************************
E3C1 Call Zero $F600 routine <E3B4>
E3C4 Copy Datetime (Creation)
E3C6 to my variables
E3D2 Loop until done ->E3C6
E3D4 Did he give Datetime (Creation)?
E3D5 Yes, carry on ->E3E2
E3D7 No, then use
E3D9 System Datetime instead (BF90)
E3E2 If Storage type is $00, $01, $02 or $03
E3E4 force it to $10
E3E6 else use a $D9
E3EC Find File name (FEB3)
E3EF OR Storage type to name length (D70W)
E3F2 Store Type/Length (FE5F)
E3F5 Isolate name length
E3F9 Copy File name to File Entry Buffer (FEB3)
E407 Copy caller's Access Byte
NOTE: This should be validity checked!!!
E40F and copy File type
E414 ----
E415 and AUX_TYPE
E41E Copy Version and Min Version (0,0) (FDF0)
E421 constants to entry (FE7B)
E42A Indicate 1 Block used
E44F Copy Directory Header Block number (FE5A)
E43E Is this a Seedling file?
E440 Yes ->E479
E442 No, Directory file - Build Header in $F600
E444 Copy completed Directory entry (FE5F)
E447 to $F600 buffer first (DC04)
E44B Loop until done ->E444
E44D Make Storage type $E in Header itself
E452 Put "HUBSTOR" (Author) in Reserved area
E45A and Version, Min Version, Access, (FDF0)
E45D Entry-length, File count and (DC20)
E460 Parent pointer from constants
E461 Loop until done ->E454
E465 Copy Parent Block entry number (FE5C)
E46C Loop until done ->E465
E46E Copy Parent entry Length (FE51)
E476 EOP = $200 (FE75)
E479 Allocate a new disk block <EAB6>
E47C error? ->E4B5

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

ADDR DESCRIPTION/CONTENTS

E407 Store it in key pointer of entry (FE70)
E408 and in BLKNUM for I/O
E40F Write zeroed (or DIR HDR) key block <E6CA>
E41B error? ->E4B5
E43D Bump parent's file count (FE53)
E495 Go update directory <E4B6>
E498 error? ->E4B5
E49A Checkpoint Volume Bit Map and exit. ->EB93

E49D ********** POINT $48/49 AT DIRECTORY ENTRY ********************
E49D $48/$49 --> Entry
E4A1 Skip link pointers (+4)
E4A3 File entry number counter (FE5E)
E4A6 ---
E4A7 Skip to proper entry
E4AB Add entry length (FE51)
E4AF (bump MSG)
E4B3 [store LSB]
E4B5 RETURN

E4B6 ********** UPDATE DIRECTORIES(S) ****************************
E4B6 System date available? (BF90)
E4B9 no, forget it ->E4C6
E4BD yes, copy to last modified date field (BF90)
E4C6 turn on BUBIT (backup) if appropriate (FE7D)
E4CF set DEVNUM of parent (FE59)
E4D4 and BLKNUM (FE5C)
E4DF reread DIR block containing entry <EB6E>
E4E0 error? ->E4B5
E4E3 Point to proper entry in buffer <E49D>
E4EB Copy constructed entry to buffer (FE5F)
E4E6 Is this block the DIR HDR block?
E501 no, write back new entry <EB6A>
E504 error? ->E4B5
E510 and then read DIR HDR block <EB6E>
E513 error? ->E4B5
E515 in any case..
E517 copy back update file count to HDR (FE53)
E520 and ACCESS Byte (with Backup) (FE5B)
E526 write back HDR block <EB6A>
E529 error? ->EB93
E52B is this the VOL DIR? (DC04)
E532 yes, all done -- exit ->E5A1
E534 no, subdirectory... (DC7)
E537 get parent pointer
E53E get parent entry no... (DC29)
E54A and entry len (DC2A)
E54D read parent DIR block <EB6E>
E54F find entry for this subdirectory <E49D>
E552 system date available? (SP90)
E555 no >>E564
E557 yes,
E55B copy system date/time to... (BP90)
E55E modified date/time in entry
E564 write it back <LibEA
E567 error? >>E563
E56B BLENUM = HDR block number
E574 same block we have now?
E578 yes, go back and date stamp >>E52B
E57A no,
E57E read HDR block <E8EE>
E581 and go back to date stamp parent DIR >>E52B
E583 error? then exit

E584 ********** NOT ProDos VOLUME ERROR ***********************
E584 ___
E587 RETURN

E588 ********** IS THIS ProDos VOLUME? ***********************
E588 Does previous block ptr = $7? (DC00)
E586 no, not a ProDos volume >>E584
E598 else, (DC04)
E59D does VOL DIR's STORAGE TYPE = $E or $F?
E59F no, error >>E584
E5AF else, ok
E5B2 RETURN

E5A3 ********** GET FILE ENTRY ***********************
E5A3 follow path to it's end <E5B6>
E5A6 error? >>E585
E5AB copy file entry
E5B3 and exit
E5B5 RETURN

E5B6 ********** FOLLOW PATH TO A FILE ***********************
E5B6 get base dir's data <E73A>
E5B9 error? >>E60D
E5BB another subdirectory in the path? >>E585
E5BD no, at end of path (E635)
E5CB $4B/49999999 (HDR)
E5CC copy part of HDR to file entry
E5D2 file type = $F (Directory) (FDE8)
E5D5 BLOCK = 2 (PE5F)
E5D8 No. blocks used = 4
E5D9 EOF = $8080

E5DD TYPE = subdirectory ($D0)
E5E2 return to caller
E5E4 RETURN

*** SCAN DIRECTORY FOR FILE ***
E5E5 indicate no free entry found as yet
E5EA signal in HDR block
E5EB zero count of names examined
E5F0 find name in block <E6E3>
E5F3 got it! >>E65A
E5F5 not yet, how many entries expected? (PE98)
E5FB less entry no. I just searched (FE97)
E5FD more file entries left to search? >>E69F
E60B no, directory error
E60D ---
E60E RETURN

E60F yes, update entries left counter (FE98)
E615 back to first buffer page (249)
E617 check next block pointer (DC02)
E61F if zero, directory error >>E60B
E621 BLENUM = next directory block
E628 read next block <EBEE>
E62B no errors, loop back for more >>E5EB
E62D exit if error

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

E657 file not found error
E659 RETURN

*** FOUND FILE ENTRY ***
Beneath Apple ProDOS Supplement

ProDOS MLI -- V1.1.1  -- 18 SEP 84
NEXT OBJECT ADDR: E65A

ADDR  DESCRIPTION/CONTENTS

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  >>E651
E668  copy key block no...
E66D  to BLKNUM
E670  and to current DIR block no (F65A)
E67A  go read key block of subdirectory <EBEE>
E67D  error?  >>E6A3
E682  new file count (F698)
E68B  check minimum version (DC21)
E69B  too new?  >>E6A1
E696  count bits in reserved field of DIR hdr
E697  ---  >>E69A
E699  ---
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  <E6A8>
E6A8  and go scan for next level  >>E5E5

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

E6AB  Copy;
E6AD  VERSION, VERSION, MIN_VERS, ACCESS, (DC1C)
E6B0  ENTRY_LEN, ENTRIES_PER_BLK, FILE_COUNT (F64A)
E6B6  volume directory? (DC04)
E6BD  if so, exit now  >>E6C8
E6C1  else, copy PARENT_POINTER, (DC27)
E6CA  PARENT_ENTRY_NO, and PARENT_ENTRY_LEN (F646)
E6CA  RETURN

E6CB  ********* SAVE DIR ENTRY N. & BLOCK ******************

E6CB  compute entry number (F652)
E6D4  save it (F65E)
E6D9  and the block it's in (F65C)
E6E2  exit

E6E1  ********* SEARCH ONE DIR BLOCK FOR FILE *************

E6E3  get entries in this block (F652)
E6E9  $4B/$49  -->  first entry (E635)
E6F0  ---
E6F2  skip HDR?  >>E727
E6F4  no, non empty entry?
E793 ****** END BASE DIRECTORY ****************************
E794 ---
E795 get the &PIXPTR (BF9A)
E796 &old qualified pathname? (FEBB)
E797 no >> E798
E799 yes, no old PIXPTR anymore
E79E save old prefix index (FEBB)
E7A0 DBVNUM=20 (BF30)
E7A4 ---

*** SCAN VCB’S FOR A MOUNTED VOLUME ***
E7A6 scan (D900)
E7A8 get one >> E787
E788 else, go to next VCB
E784 no mounted vols? remount them >> E888

*** END LAST DIR IN PREFIX OR TOL DIR ***
E787 store name length (FEB8)
E789 same name as in pathname? (D780)
E78D no >> E8A8
E7CB save VCB index (FE91)
E7CD DEVNUM= VCB’s unit no. (D916)
E7D4 BLOCK = 2 (read VOLDIR if no old PREFIX)
E7DC get old prefix index (FEBB)
E7DF ---
E7EE accumulate a new index (FEBA)
E7E1 previous prefix? >> E7F5
E7E6 find last name in prefix (D780)
E7EB read prefix directory instead of vol dir (FEA0)
E7F5 read block <EBEB>
E7F8 error? >> E800
E7FA is this the right directory? <EB9E>
E7FD no >> E800
E7FF yes, exit!

*** NOT THERE, REMOUNT ALL VOLS ***
*** AND CHECK THEM ***
E800 open files? (FE91)
E806 yes, give up now >> E821
E808 else, (FEBB)
E809 put back old prefix length (FEBA)
E80E copy DEVLIST 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

E825 ---
E828 search for device in device table (FECA)
E830 device not found >> E821
E832 when found, make it active device (BF30)
E837 remove it from table (FECA)
E83A find its VCB <E876>
E83D not found? >> E863
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 <EBEE>
E857 error? >> E816
E859 mount volume on VCB <E8C4>
E85C error? >> E816
E85E is this his chosen volume? <EB9E>
E861 no, try again >> E816
E863 yes, exit

E864 ********** COPY GLBL DEVLIST TO MY TABLE *************
E864 start with last device (BF31)
E867 get a unit number (BF32)
E86C copy it to device table (FECA)
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? >> E88B
E883 is it, save VCB index (FE91)
E886 and exit normally
E887 RETURN

E88B 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 >> E99B
E89A else, all is well -- return empty VCB
E99B VCB table full error
E99D RETURN
### ProDOS MLI -- V1.1.1 -- 18 SEP 84

#### ADDR DESCRIPTION/CONTENTS

<table>
<thead>
<tr>
<th>ADDR</th>
<th>Description/Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>E89E</td>
<td>********** COMPARE DIR NAME WITH PATH_LVL **********</td>
</tr>
<tr>
<td>E89E</td>
<td>---</td>
</tr>
<tr>
<td>E8AE</td>
<td>check DIR type (DC94)</td>
</tr>
<tr>
<td>E8AF</td>
<td>VOL DIR or SUB DIR?</td>
</tr>
<tr>
<td>E8AA</td>
<td>neither &gt;&gt;E8B1</td>
</tr>
<tr>
<td>E8AC</td>
<td>yes..</td>
</tr>
<tr>
<td>E8AC</td>
<td>store len of its name (FEB8)</td>
</tr>
<tr>
<td>E8AF</td>
<td>and go on &gt;&gt;E8B6</td>
</tr>
<tr>
<td>E8B1</td>
<td>error exit</td>
</tr>
<tr>
<td>E8B2</td>
<td>RETURN</td>
</tr>
<tr>
<td>E8B3</td>
<td>compare directory names (DC04)</td>
</tr>
<tr>
<td>E8B9</td>
<td>no match? &gt;&gt;E8B1</td>
</tr>
<tr>
<td>E8C2</td>
<td>they match! exit</td>
</tr>
<tr>
<td>E8C3</td>
<td>RETURN</td>
</tr>
<tr>
<td>E8C4</td>
<td>********** MOUNT NEW VOLUME **********</td>
</tr>
<tr>
<td>E8C4</td>
<td>volume mounted? (FE91)</td>
</tr>
<tr>
<td>E8CA</td>
<td>no, continue &gt;&gt;E8D1</td>
</tr>
<tr>
<td>E8CC</td>
<td>yes, same one as one wanted? &lt;E929&gt;</td>
</tr>
<tr>
<td>E8CF</td>
<td>if so exit, else fall thru &gt;&gt;E928</td>
</tr>
<tr>
<td>E8D1</td>
<td>********** SET UP VCB FROM VOLDIR **********</td>
</tr>
<tr>
<td>E8D1</td>
<td>zero out VCB</td>
</tr>
<tr>
<td>E8DC</td>
<td>is this a ProDOS volume? &lt;E808&gt;</td>
</tr>
<tr>
<td>E8DF</td>
<td>no -- exit &gt;&gt;E928</td>
</tr>
<tr>
<td>E8E1</td>
<td>duplicate vol in VCB's? &lt;E94A&gt;</td>
</tr>
<tr>
<td>E8E4</td>
<td>yes -- exit with that one instead &gt;&gt;E927</td>
</tr>
<tr>
<td>E8E6</td>
<td>get new volume's name length (DC04)</td>
</tr>
<tr>
<td>E8ED</td>
<td>add to VCB index (FE91)</td>
</tr>
<tr>
<td>E8F1</td>
<td>and copy to VCB name field in empty VCB (DC04)</td>
</tr>
<tr>
<td>E8FC</td>
<td>store in VCB name len field (D900)</td>
</tr>
<tr>
<td>E8FF</td>
<td>copy DEVNUM to VCB unit field (BF38)</td>
</tr>
<tr>
<td>E905</td>
<td>copy total blocks to VCB (DC29)</td>
</tr>
<tr>
<td>E911</td>
<td>copy block no. of vol dir to VCB</td>
</tr>
<tr>
<td>E91B</td>
<td>copy bit map block no. to VCB (DC27)</td>
</tr>
<tr>
<td>E927</td>
<td>exit</td>
</tr>
<tr>
<td>E928</td>
<td>RETURN</td>
</tr>
<tr>
<td>E929</td>
<td>********** COMPARE VOL NAMES TO MAKE **********</td>
</tr>
<tr>
<td>E929</td>
<td>***** SURE THEY MATCH *****</td>
</tr>
<tr>
<td>E929</td>
<td>get length (DC04)</td>
</tr>
<tr>
<td>E92E</td>
<td>same in VCB? (D900)</td>
</tr>
<tr>
<td>E931</td>
<td>no &gt;&gt;E941</td>
</tr>
<tr>
<td>E934</td>
<td>yes, add len to VCB index to point at (FE90)</td>
</tr>
<tr>
<td>E937</td>
<td>last char of name in VCB (FE90)</td>
</tr>
</tbody>
</table>

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

#### ADDR DESCRIPTION/CONTENTS

<table>
<thead>
<tr>
<th>ADDR</th>
<th>Description/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>E93E</td>
<td>compare names (D900)</td>
</tr>
<tr>
<td>E941</td>
<td>SEC if no match</td>
</tr>
<tr>
<td>E948</td>
<td>CLC if match</td>
</tr>
<tr>
<td>E949</td>
<td>RETURN</td>
</tr>
<tr>
<td>E94A</td>
<td>********** LOOK FOR DUPLICATE VOL **********</td>
</tr>
<tr>
<td>E94A</td>
<td>start with first VCB</td>
</tr>
<tr>
<td>E94C</td>
<td>---</td>
</tr>
<tr>
<td>E94D</td>
<td>this VCB has same name? &lt;E929&gt;</td>
</tr>
<tr>
<td>E950</td>
<td>no &gt;&gt;E961</td>
</tr>
<tr>
<td>E952</td>
<td>yes, files open? (D911)</td>
</tr>
<tr>
<td>E955</td>
<td>yes &gt;&gt;E968</td>
</tr>
<tr>
<td>E959</td>
<td>no, mark VCB empty (NAME=0) (D900)</td>
</tr>
<tr>
<td>E95C</td>
<td>UNIT=0) (D910)</td>
</tr>
<tr>
<td>E95F</td>
<td>and exit with no error &gt;&gt;E969</td>
</tr>
<tr>
<td>E961</td>
<td>else,</td>
</tr>
<tr>
<td>E963</td>
<td>bump to next VCB</td>
</tr>
<tr>
<td>E967</td>
<td>and loop &gt;&gt;E94C</td>
</tr>
<tr>
<td>E969</td>
<td>exit no errors</td>
</tr>
<tr>
<td>E96A</td>
<td>RETURN</td>
</tr>
<tr>
<td>E96B</td>
<td>save flag (FE55)</td>
</tr>
<tr>
<td>E96E</td>
<td>and VCB index of duplicate vol (FE86)</td>
</tr>
<tr>
<td>E971</td>
<td>exit with error</td>
</tr>
<tr>
<td>E972</td>
<td>RETURN</td>
</tr>
<tr>
<td>E973</td>
<td>********** SEE IF A QUANTITY OF FREE BLOCKS AVAILABLE ON VOLUME **********</td>
</tr>
<tr>
<td>E973</td>
<td>any free blocks counted in VCB? (FE91)</td>
</tr>
<tr>
<td>E97C</td>
<td>yes &gt;&gt;E9D0</td>
</tr>
<tr>
<td>E97E</td>
<td>*** COMPUTE VCB FREE BLOCK COUNT ***</td>
</tr>
<tr>
<td>E97E</td>
<td>no, how many bit maps are there? &lt;EA22&gt;</td>
</tr>
<tr>
<td>E9B1</td>
<td>save it (less 1) (FE9C)</td>
</tr>
<tr>
<td>E9B6</td>
<td>zero scratch (will count free blocks) (FE86)</td>
</tr>
<tr>
<td>E9BC</td>
<td>no block found yet</td>
</tr>
<tr>
<td>E991</td>
<td>checkpoint bit map buffer &lt;EB93&gt;</td>
</tr>
<tr>
<td>E994</td>
<td>error? &gt;&gt;E9E4</td>
</tr>
<tr>
<td>E999</td>
<td>BLKNUM = bit map pointer (D91A)</td>
</tr>
<tr>
<td>E9A3</td>
<td>read block buffer to &lt;EBEE&gt;</td>
</tr>
<tr>
<td>E9A6</td>
<td>error? &gt;&gt;E9E4</td>
</tr>
<tr>
<td>E9A8</td>
<td>count free blocks marked &lt;EB95&gt;</td>
</tr>
<tr>
<td>E9AB</td>
<td>drop no. remaining to do (FE9C)</td>
</tr>
<tr>
<td>E9AE</td>
<td>none left? &gt;&gt;E9B9</td>
</tr>
<tr>
<td>E9B0</td>
<td>some, BLKNUM = BLKNUM + 1</td>
</tr>
<tr>
<td>E9B6</td>
<td>go process that &gt;&gt;E9A3</td>
</tr>
</tbody>
</table>
E9B9 did we find a free bit? (FE91)
E9BF 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
E9E8 counting one bits <EA12>
E9F7 ----
E9FA found free block already? (FE9B)
E9FD if so -- done >>EA11
E9FF any blocks found yet? (FE86)
EA05 no >>EA11
EA07 yes, compute total no. of bitmap blocks <EA22>
EA08 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 ----
EA3F 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.
EA48 save it (FE9B)
EA4F divide block no. by 8 (FE9C)

EA52 description/contents
EA58 next object addr: EA52
EA5E mem
EA61 curring byte offset as remainder
EA64 rem byte offset (FEA2)
EA67 rem quotient/2 into block index (FE9C)
EA69 remamber which page in that block (FEA4)
EA6E yes bit map block (after checkpoint) <EB64>
EA71 if quot >>EA81
EA74 are we at proper block of bitmap yet? (FEA9)
EA76 in 1:4 >>EA87
EA7E DERR: checkpoint <EB93>
EA82 remor >>EA81
EA85 indicate block wanted in VCB (FE9C)
EA87 get num of bitmap (FEA6)
EA8A get actual block directly <EA4>
EA8D get quot >>EA81
EA8F get byte offset into page (FEA2)
EA92 remamber page? (FEA4)
EA95 and bit pattern to set (FE9B)
EA9A rem quot ?? >>EA9A
EA95 the turn bit on in page1 (DB00)
EA98 continue >>EA90
EA9B exist bit on in page0 (DA00)
EA9F free bitmap needs checkpoint
EA91 last block freed (FEC2)
EA92 DERR: normally
EA95 RETURN

EA96 ******** bitmap error

EA96 *** FIND A FREE DISK BLOCK AND **********

EA99 *** AND ALLOCATE IT **********

EA9B DERR

EAAC read bitmap <EB64>
EAC8 por? >>EABD
EACB last page? (FEA3)
EACC rem last page of bitmap for free block (DA00)
EADD rem tm page 1 of buffer (FEA4)
EADE remro page offset (FEA3)
EADF rem 2nd page too (DB00)
EAEE rem page (FEA3)
EAFF nextblock <EB42>
EA0D continue >>EABB
EA02 por exit
EA05 by
EA08 por byte index (FEA2)
EA07 rem the combination of page no. and (FEA3)
EA0C rem offset left 3 bits to make (FE87)
EA01 set for bit position.
end on buffer page ... (FEA4)
oad bit pattern from page 0... (DB00)
page 1 (DA00)
ProDOS MLI -- V1.1.1 -- 18 SEP 84  NEXT OBJECT ADDR: EBA3

ADDR   DESCRIPTION/CONTENTS

EBA4  save DEVNUM (FEA6)
EBA7  copy block offset wanted (FE91)
EBA9  get bitmap block = BLOCK PTR + BLOCK OFFSET (D91A)
EBBF  set up read command

*** READ OR WRITE BITMAP ***

EBC1  save I/O command
EBC7  device = bitmap device (FEA6)
EBCD  block = bitmap block (FEA7)
EBDA  do the I/O <<EBF5>>
EBDF  restore old DEVNUM (BP30)
EBE2  ok? >>EBE5
EBE4  no, error exit
EBE5  RETURN

EBE6  save I/O command
EBCD  block = bitmap block (FEA7)
EBDA  do the I/O <<EBF5>>
EBDF  restore old DEVNUM (BP30)
EBE2  ok? >>EBE5
EBE4  no, error exit
EBE5  RETURN

EBE9  save I/O command
EBF2  where is my buffer? (E635)
EBF5  save flags
EBF6  and disable
EBF9  Set low byte of Buffer pointer
EBFB  to zero
EBFD  Initialize Global Page System error to 0 (BF0F)
ECBO  set I/O transfer occurred flag
EC05  set unit to do I/O on (BF30)
EC0F  error? >>EC12
EC0F  no errors, restore things and exit
EC11  RETURN

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

EB6A  READ BITMAP BLOCK ******************************

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)
EB9A  (page number) (D91C)
EB91  exit
EB92  RETURN

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

EB93  ----
EB94  needs checkpoint? (FEA5)
EB97  no >>EB92
EB99  yes, write it <<EB6A>
EB9C  error? >>EB92
EB9E  doesn't need checkpoint now
EBAE  exit
** *** NEED DIFFERENT DATA BLOCK *** **

EC81 copy storage type (D807)  
EC82 old data block needs writing? (D808)  
EC83 no >>EC95  
EC90 yes, do so >>EC94  
EC93 error? >>EC9E  
EC95 see if new mark is outside the range of (FE92)  
EC98 the current index block (D814)  
EC97 yes >>EC9C  
EC99 yes >>EC9C  
ECA0 no, same index block (FE96)  
ECA1 check storage type  
ECA2 sapling or tree are ok >>ED2D  

** *** NEED TO CHANGE DATA BLOCKS *** **

EC37 does old index block need dumping? (D808)  
EC3C no >>EC33  
EC3E yes, do so >>EC3A  
EC31 error? >>EC3E  
EC33 check storage type (FE96)  
EC36 tree file?  
EC38 yes >>ED80  
ECD1 no, sapling (FEAC)  
ECD3 is position in first index block?  
ECD4 no, need master index, subindex and data >>ED46  
ECD5 yes, first index, reset flags >>EDAF  
ECD7 if so, see if in first block >>ECB3  

** *** S P A L L I N G *** **

EC3A no, sapling, read its only index block >>EE3B  
EC3D error? >>EC3E  
EC3F set block no. of index block  
EC3A and continue below >>ED2D  
EC3A error exit  
EC3F return
### Beneath Apple ProDOS Supplement

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ED00</td>
<td>reset flags &lt;EDAF&gt;</td>
</tr>
<tr>
<td>ED03</td>
<td>read master index block &lt;EE3B&gt;</td>
</tr>
<tr>
<td>ED06</td>
<td>error? &gt;&gt;ECPE</td>
</tr>
<tr>
<td>ED08</td>
<td>make index into block from (FEAC)</td>
</tr>
<tr>
<td>ED0B</td>
<td>MSR_of_position/2</td>
</tr>
<tr>
<td>ED11</td>
<td>is there a subindex there?</td>
</tr>
<tr>
<td>ED13</td>
<td>yes! &gt;&gt;ED20</td>
</tr>
<tr>
<td>ED19</td>
<td>no, fall thru to make one</td>
</tr>
</tbody>
</table>

### *** GET NEW INDEX BLOCK ***

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ED1B</td>
<td>need an index and data block</td>
</tr>
<tr>
<td>ED1D</td>
<td>go allocate them &gt;&gt;ED46</td>
</tr>
<tr>
<td>ED20</td>
<td>set up block no. of subindex</td>
</tr>
<tr>
<td>ED28</td>
<td>read it &lt;EE1D&gt;</td>
</tr>
<tr>
<td>ED2B</td>
<td>error? &gt;&gt;ECPE</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ED2D</td>
<td>make block no. out of position (FEAC)</td>
</tr>
<tr>
<td>ED36</td>
<td>use as an index to examine index block</td>
</tr>
<tr>
<td>ED3B</td>
<td>entry</td>
</tr>
<tr>
<td>ED3E</td>
<td>if its zero...</td>
</tr>
<tr>
<td>ED42</td>
<td>need new data block</td>
</tr>
<tr>
<td>ED46</td>
<td>set flags for what to allocate (FE92)</td>
</tr>
<tr>
<td>ED4F</td>
<td>new index block being created?</td>
</tr>
<tr>
<td>ED51</td>
<td>zero data block in any case &lt;ED67&gt;</td>
</tr>
<tr>
<td>ED54</td>
<td>if not index block that's it &gt;&gt;ED89</td>
</tr>
<tr>
<td>ED56</td>
<td>else,</td>
</tr>
<tr>
<td>ED5D</td>
<td>zero out index block I/O buffer</td>
</tr>
<tr>
<td>ED64</td>
<td>and continue &gt;&gt;ED89</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ED67</td>
<td>************ ZERO OUT DATA BLK I/O BUFFER ************</td>
</tr>
<tr>
<td>ED69</td>
<td>---</td>
</tr>
<tr>
<td>ED70</td>
<td>zero both pages of buffer</td>
</tr>
<tr>
<td>ED78</td>
<td>RETURN</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ED79</td>
<td>************ READ FILE DATA BLOCK ************</td>
</tr>
<tr>
<td>ED79</td>
<td>set block no. LSB</td>
</tr>
<tr>
<td>ED7B</td>
<td>copy MSB drom index entry</td>
</tr>
<tr>
<td>ED7F</td>
<td>---</td>
</tr>
<tr>
<td>ED81</td>
<td>read new data block &lt;EE04&gt;</td>
</tr>
<tr>
<td>ED84</td>
<td>error? &gt;&gt;EDAF</td>
</tr>
<tr>
<td>ED86</td>
<td>reset block allocation flags &lt;EDAF&gt;</td>
</tr>
</tbody>
</table>

### *** GOT DATA BLOCK WANT ***

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ED89</td>
<td>---</td>
</tr>
<tr>
<td>ED90</td>
<td>save previous mark in my</td>
</tr>
<tr>
<td>ED96</td>
<td>set new mark in the PCB (w)</td>
</tr>
<tr>
<td>EDA1</td>
<td>($4A/$4B --&gt; data block bu</td>
</tr>
<tr>
<td>EDA3</td>
<td>%4C/$4D --&gt; start of the ED</td>
</tr>
<tr>
<td>EDA5</td>
<td>the data block buffer while</td>
</tr>
<tr>
<td>EDA8</td>
<td>the mark.</td>
</tr>
<tr>
<td>EDAE</td>
<td>exit</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>EDAF</td>
<td>************ RESET BLOCK ALLOC (ffer)</td>
</tr>
<tr>
<td>EDAF</td>
<td>get flags (FE92)</td>
</tr>
<tr>
<td>EDB5</td>
<td>turn off low 3 bits (alloc)</td>
</tr>
<tr>
<td>EDB7</td>
<td>blocks to file) (DA88)</td>
</tr>
<tr>
<td>EDBA</td>
<td>RETURN</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>EDBB</td>
<td>************ SET DIR FILE POSIT</td>
</tr>
<tr>
<td>EDBB</td>
<td>DIR file?</td>
</tr>
<tr>
<td>EDBD</td>
<td>yes! &gt;&gt;EDC4</td>
</tr>
<tr>
<td>EDBF</td>
<td>no, bad storage type erro</td>
</tr>
<tr>
<td>EDC1</td>
<td>go to SYSSERR &lt;SF99&gt;</td>
</tr>
<tr>
<td>EDC4</td>
<td>else, get page distance (div)</td>
</tr>
<tr>
<td>EDC7</td>
<td>make it into blocks (div)</td>
</tr>
<tr>
<td>EDC8</td>
<td>new position beyond old?</td>
</tr>
<tr>
<td>EDD1</td>
<td>yes &gt;&gt;EDE1</td>
</tr>
<tr>
<td>EDD3</td>
<td>else, use previous mark</td>
</tr>
<tr>
<td>EDD5</td>
<td>copy to BLKNUM &lt;EDEF&gt;</td>
</tr>
<tr>
<td>EDD8</td>
<td>error? &gt;&gt;EDFE</td>
</tr>
<tr>
<td>EDDA</td>
<td>count it (FED0) ( divides by 2)</td>
</tr>
<tr>
<td>EDDD</td>
<td>more to skip? &gt;&gt;ED33</td>
</tr>
<tr>
<td>EDDF</td>
<td>no, got it &gt;&gt;ED89</td>
</tr>
<tr>
<td>EDE1</td>
<td>use next_block pointer in</td>
</tr>
<tr>
<td>EDE3</td>
<td>copy to BLKNUM &lt;EDEF&gt;</td>
</tr>
<tr>
<td>EDE6</td>
<td>error? &gt;&gt;EDFE</td>
</tr>
<tr>
<td>EDE8</td>
<td>count it (FED9)</td>
</tr>
<tr>
<td>EDED</td>
<td>more to skip &gt;&gt;EDE1</td>
</tr>
<tr>
<td>EDEF</td>
<td>got it now! &gt;&gt;ED89</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>EDEF</td>
<td>************ COPY LINK TO BLKNUM ***</td>
</tr>
<tr>
<td>EDEF</td>
<td>copy block number link</td>
</tr>
<tr>
<td>EDF1</td>
<td>to BLKNUM</td>
</tr>
<tr>
<td>EDF4</td>
<td>if non zero,</td>
</tr>
<tr>
<td>EDF5</td>
<td>then go read block. &gt;&gt;E90</td>
</tr>
<tr>
<td>EDFC</td>
<td>else, EOF error ***</td>
</tr>
<tr>
<td>EDFE</td>
<td>return</td>
</tr>
<tr>
<td>EDFI</td>
<td>return</td>
</tr>
</tbody>
</table>
just read to PCB

command
49 buffer
K <E61>

a PCB as current index

KEY INDEX BLOCK **********************

command
1/O >>E3D

BY INDEX BLOCK **********************

command
save command
key block in FCB (FE92)
buffer
CK ***

nd
(D800)
zero block number
next to read/write block (D801)
AND DO FILE BLOCK 1/O ***

ptr in zero page)

pointer
from PCB (D801)
buffer has occurred flag
from DEVNUM (BF38)
buffer occurred yet <DEDA>

--- 18 SEP 84 NEXT OBJECT ADDR: EDF

FILE BLOCK **********************

----- 18 SEP 84 NEXT OBJECT ADDR: EDF

DESCRIPTION/CONTENTS

FILE BLOCK **********************

buffer to read
D command
49 buffer
K <E61>

--- 18 SEP 84 NEXT OBJECT ADDR: EDF

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

--- DESCRIPTION/CONTENTS ---

EEE4 zero out unused PCB
EEE5 copy file ID fields to PCB
EEF2 (DEVNUM, DIR HDR BLK, DIR BLK, (F9E2)
EEF5 DIR ENTRY NO.)
EF06 isolate storage type (F95F)
EF08 and copy to PCB (D9B7)
EF08 get access (F97D)
EF10 DIR file?
EF12 no >>EF16
EF14 yes, we are only reading (I hope)
EF16 update access flag in PCB (D9B9)
EF18 write protected? >>EF22
EF1D no, another PCB open on this file? (F9E7)
EF20 yes, no touchie >>EED4
EF22 This line left over from version 1.0.11 (F97C)
EF27 Now always jumps over error exit. >>EF2D
EF29 if bad, unsupported version error
EF2C RETURN

--- REASON/CONTENTS ---

EF94 else:
EF97 entry read key block to I/O buffer <E9E4>
EF99 bump? >>EF7D
EF9F ind: open file count in VCB (F9E9)
EFA7 put date files are open in VCB (D911)
EFA8 exit.X:NUM in caller's perm list (F9E2)
EFA2 RETN with no errors

--- FIND A PCB ---

EFB3 clear:
EFBE --- old flags and index byte
EFBF four:
EFC2 yes: a free PCB yet? (F9E3)
EFCC no, >>EF09
EFCF save:
EFD2 flag index to free PCB (F9E2)
EFD7 and: that we found one
EFD9 --- skip this PCB >>EF77
EFD9 comp:
EFE2 is file ID's to see if this PCB (D9B0)
EFE5 no open on the requested file. (F9E8)
EFE8 ind: match? >>EF77
EFE8 write PCB already open on file (FE97)
EFE3 if te enabled? (D9B9)
EFE5 else:
EFE6 RETN, allow multiple open access to file >>EF77
EFE6 RETN, error exit

EFE7 return
EFE8 bumpn index to start of PCB
EFE9 and: to next PCB
EFFF wend loop >>EFBE
F000 RETN done, exit normally

---*********** ML/READ CALL ***********---

---*********** ML/READ CALL ***********---
ProDOS MLI -- V1.1.1 -- 18 SEP 84

**ADDR** DESCRIPTION/CONTENTS

---

F019 LENGTH = EOF - current mark (D815)
F031 are we already at EOF? (FEDA)
F034 no >>F046
F036 yes, EOF error

F03B else, zero length request? (FEDA)
F041 no >>F046
F043 yes, set mark and exit >>F0P9

F046 validity check data buffer <F062>
F049 no good? >>F038
F04B ok, get storage type for file <F210>
F04E standard kind of file?
F050 yes >>F055
F052 no, DIR file >>F1BB

F055 else, set mark (to read proper buffers) <EC48>
F058 error? >>F038
F05A set up buffer indexing <<F110
F05D move all that can be moved out of data buff <<F13A
F060 newline or len=0: exit now! >>F043
F062 newline enabled? continue block by block >>F055
F064 at least 1 block's worth left to be read? (FEAE)
F066 if not, never mind >>F055
F06A if so, store block count wanted (FEAF)
F06D get FCB flags <F066>
F070 data block modified?
F072 yes, continue block by block for now >>F055

--- FAST DIRECT READ ROUTINE ***

F074 signal no read occurred yet <<FEB2
F077 read directly into caller’s data buffer
F07F set mark/read data block to caller's buff <EC48>
F082 error? >>F0ED
F084 bump buffer pointer to next location
F088 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)
F09F drop counter of multi-blocks (FEAF)
F0A2 and keep on >>F081
F0A4 end of multi-block read, put ptrs back <<F1AD
F0A7 more to read? (FEAD)
F0AD no, exit through finish-up >>F0F9
F0AF yes, conventional block by block read then >>F055

---

**ADDR** DESCRIPTION/CONTENTS

---

F0B1 crossed index block? go do set mark >>F07F
F0B3 make index block offset from mark (FEAC)
F0BC BLKNUM = next block in index block
F0C2 zero entry?
F0CA if so, no direct read can occur until next (FEB2)
F0CD set-mark/read >>F0D2
F0CF get MSB of BLKNUM
F0D2 (put index ptr back)
F0D6 finish setting BLKNUM MSB
F0D8 if no read occurred within setmark, (FEB2)
F0DB go back to setmark call >>F07F
F0DF disable
F0E0 do i/o to caller's buffer directly
F0E4 do block I/O directly <FEDA>
F0E7 error? >>F0EC
F0EA go back for more >>F084

*** ERROR CLEANUP ***

F0EC ---
F0ED ---
F0EE set buffer ptrs/VCB <<F1AD
F0F2 ---
F0F3 finish up I/O <<F0F9
F0F7 exit with error
F0FB RETURN

F0F9 ******** I/O FINISH UP ******************************************

F0F9 ---
F0FC return actual length read in caller's list (FEDA)
F10D and exit by setting new mark >>EC48

---

**F110 ******** 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? (D81F)
F123 no, CLC >>F12F
F125 yes, SEC
F126 copy newline mask (FEB1)
F129 and newline character (D80A)
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
Beneath Apple ProDOS Supplement

ProDOS MLI -- V1.1.1
------------------------
18 SEP 84
NEXT OBJECT ADDR: F139

PRODOS MLI -- V1.1.1
------------------------
18 SEP 84
NEXT OBJECT ADDR: F1AC

F13A ********** COPY F13A

****** TO DRAW I/O BLOCK BUFF **************

EXITS IF: ON BUFFER

NEXT BLOCK IS NEEDED

ON EXIT: OVERFLOW FLAG SET IF DONE

OVERFLOW ZERO IF NEXT BLOCK NEEDED

F13A ---

F13B partial page

F13D no, any full page move? >>F145

F140 no, read commands left? (PEAE)

F142 yes, drop MSBlock >>F194

F145 ---

F146 request length (PEAE)

F147 copy one byte

F148 end of request SAC >> $4E

F14D no, newline ended chunk? >>F168

F14F ---

F150 no, loop for ...

F151 end of page, done >>F166

F152 bump new mark

F153 finished first? (PEAE)

F156 if so, continue page of block buffer?

F166 no, need another >>F146

F168 another page of block from disk >>F197

F16B no >>F1B7 in request length? (PEAE)

F16E more in this block?

F170 no, on last block page? >>F176

F174 yes, drop request

F179 back up to newest len by one page (PEAE)

F17A go copy next byte again

F17D check for new

F185 not it, neverline

F187 else, were we mind >>F14F

F188 no >>F194 done with page?

F18A yes, bump point

F18C and mark (PEACE)

F194 set overflow

F197 update mark

F19C bump request SB (PEAA)

F19D update count count if necessary

F1A3 point beyond CSS (PEAD)

F1A6 ---

F1A8 data in caller's buffer

F1AC and exit

F1A0 set up buffer in buffer >>F13A

F1C3 move data from ... >>F1BB

F1C6 need next block? >>F0F9

F1CB no, finish up if?

F1CC ok? exit >>F1ED

F1CD not ok, EOF error?

F1DD no, out now >>F1D2 EOF anyway? >>ED99

F1D2 yes, point beyond buffer >>ED67

F1D5 zero out data block >>DIR block with previous (D810)

F1DD dummy up an empty block block number in 1/O

F1ED return to caller

F1EE RETURN

F1F2 finish up and exit >>F0F2

F1F2 ********** COPY CALL

with to LENGTH

F1F2 copy request length

F1F4 a temporary variables for file (FEA92)

F205 pick up ACCESS

F20B exit to caller

F20C RETURN

F20D ********** POINT $4967 FOR CALLER'S **********

***** DATA BUFFER

F20D set up pointer >>92

F218 YREG --> FCB (type D807)

F21B ARREG = storage

F21E exit

F21F ********** COPY FROM PARZ END MARK *****

***** AND CC
F24C  ******** SET NEW MARK & EOF
F24C  set up indexes <F27E>
F24F  set new EOF in FCB (FEDA)
F255  and new mark (FEBD)
F25B  save new mark in scratch
F262  does mark exceed EOF? <F277>
F265  if so, we must extend FEBD
F268  save old EOF (D015)
F273  set new EOF to mark if
F277  and go if no errors >>F26D
F279  exit
F27E  subroutine to set 3 byte
F285  RETURN

F286  ***********************
***** MLI WRITE CALL
***********************
F286  copy request length =>FEB6
F287  copy file mark <F1F7>
F28D  extend EOF if needed <F27F>
F291  write access enabled? yes >>F299
F295  no, access error
F299  check status of this dev'
F29C  error? >>F29D
F29E  request length = 07 (FED7) indexes
F2A4  no >>F2A9
F2A6  yes, exit through finish:

F2A9  find caller's data
F2AC  check storage type
F2AE  if DIR file, error >>F29F
F2B0  set mark/read blocks <F5B>
F2B3  error? >>F29D
F2B5  get FCB flags <F606>
F2B8  any new blocks needed?
F2BA  no >>F31E
F2BC  yes, allocating them
F2BE  ---

Beneath Apple ProDOS Supplement

ProDOS MLI -- V1.1.1 -- 18 SEP 84
---
ADDR  DESCRIPTION/CONTENTS
---
F21F  ---

F2BF  count number of blocks needed
F2C2  store number needed (F694)
F2CB  see if the blocks are available <E973>
F2CB  no, disk full >>F2D9
F2CD  yes, get 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
F2DF  check FCB flags again <F606>
F2E4  need sub-index block?
F2E6  no >>F2ED
F2E8  yes, do go it <F3E4>
F2EB  error? >>F29D
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 (D608)
F2FB  make index block offset from mark
F307  store new block no. in index block (FEB7)
F314  and store it as current data block (F692)
F31E  set up buffer indexing <F178>
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 (FEB6)
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 $106 (F6A8)
F34B  still in same I/O block page?
F34F  yes >>F34A
F352  no, clear overflow (I/O incomplete) >>F379
any complete pages left to write? (PBAE)
no >>F369
yes, more in this page?
yes >>F362
no, first block-page?
no >>F365
readjust index
continue with full page >>F33C

---
a few bytes left to write? >>F376
no, bump data buffer by #100
and mark (PBAE)
set overflow (I/O complete) (FLAC)
store LSB of mark (PBAE)
and of request count (PEAD)
indicate data block modified (F606)
and DIR entry needs update
advance pointer into caller's buffer (PBAE)
exit

--- ADD NEW MASTER INDEX BLOCK *******************************
(MAKE A TREE FILE)
add higher level >>F31F
error? >>F3F0
get storage type (F218)
tree?
yes >>F3AA
no, add another level >>F3F1
error? >>F3F0
buy another block >>F3F8
error? >>F3F0
make offset into current index block (FLAC)
get current mark
point index to new block (F606)
also save as current data block (F922)
checkpoint bitmap & key block (E69E)
error? >>F3F0
zero out new index block
---
and exit

--- ADD NEW INDEX BLOCK ******************************
check storage type (F218)
no? >>F3F1
read key index block (E63B)
add data block >>F3AA
if error occurs
---
ADD A HIGHER INDEX LEVEL TO FILE ***
buy more block >>F43B
save? >>F437
make old key block number (D80C)
and's new block the key block (D80C)
store current index block in PCB (D86F)
in pointer to old key block
check first position of new index
error? >>F437
point bitmap and new key block (E68E)
get #? >>F437
update storage type (F218)
indicate it to next higher type (D807)
exit

--- BUY A DISK BLOCK ******************************
allocate
errata a disk block (EAB6)
get #? >>F457
indicate PCB flags (F606)
add date DIR entry needs update
--- + to blocks in use for file
exit

--- DO STATUS IF NO I/O YET ******************************
get #?
give PCB flags (F606)
if no in use? (I/O activity)
no, assume its ok >>F456
set new device (BF20)
--- STATUS CALL ***
---
Save
Save Unit Number
Ind Block Number on stack
Indicate Status call
Go to stack block 9
Res. I/O <DEDAD>
Exit
ProDOS MLI -- v1.1.1 -- 18 SEP 84

ADDR DESCRIPTION/CONTENTS

F481 **********************
| LI CLOSE CALL **********
| **********************
| **********************
   **************************
   F481 check REQ?
   F485 specific
   E ALL OPEN FILES ***
   *** CLOS
   y (FEBE)
   F487 no errors
   F48C store PCB
   get its t
   F490 if belo
   et >>F491
   F491 if belo
   et >>F492
   F492 yes, skip
   F496 no, active
   F498 no >>F499
   F49D yes, file
   F4AD error? >>F51E
   F4A0 error? >>F51E
   F4A2 no, close
   F4A4 is this a
   F4A9 yes, ignor
   F4AB no, stop
   F4AD bump FCB
   F4B3 and conti
   F4B5 when done
   F4BB end exit |
   E SPECIFIC FILE ***
   *** CLOS
   :: F526
   F4BC flush li
   F4BF error? >>F51E
   F4C1 get buffer
   F4C7 free its
   F4CA error? >>F51E
   F4CC release F
   D4 set DEV
   F4DA find VCB
   F4DD decrement
   F4E5 if all ar
   F4E6 "files o
   F4ED ***
   F4EE exit
   F4EF jump to
   F4F2 **********************
   MLI FLUSH CALL
   ****************************

   F4F2 flush specific file?
   F4F6 no >>F526
   F4F8 yes, flush it <F51E>
   F4FD set FCB index for next FCB (FE92)
   F501 if this file open? (D80E)
   F504 no >>F56B
   F506 yes, flush it <F51E>
   F509 error? >>F51B
   F51A return with error code if any (FEBE)
   F51B return

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

   F51E find buffer/VCB <E1EB>
   F521 no error? >>F530
   F523 error - exit >>F5F7
   F526 zero out close-all error
   F52B validity check REQ NUM <E1D0>
   F52E error? >>F51B
   F530 is write access allowed? (D809)
   F535 no, exit >>F513
   F537 has a write occurred since last flush? (D81C)
   F53A yes >>F543
   F53C no, <F56B>
   F53F does anything need flushing anyway?
   F541 no, then exit now >>F513
   F543 else, get FCB flags <F606>
   F546 has data buffer changed?
   F548 no >>F54F
   F54A yes, checkpoint it <EE94>
   F54D error? >>F51B
   F54F get flags again <F606>
   F552 has index buffer changed?
   F554 no >>F55B
   F556 yes, checkpoint it <EEA8>
   F559 error? >>F51B
   F55B return
   F562 copy file identifier data to my variables (D800)
   F56C set DEVNUM (BF30)
Beneath Apple ProDOS Supplement

ProDOS MLI -- V1.1.1 -- 16 SEP 84
NEXT OBJECT ADDR: F56F

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>F56F</td>
<td>BFILENAME = 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;F51B</td>
</tr>
<tr>
<td>F57E</td>
<td>copy directory header &lt;EBAB&gt;</td>
</tr>
<tr>
<td>F581</td>
<td>we are in block with this file's entry? (FE5C)</td>
</tr>
<tr>
<td>F58B</td>
<td>no &gt;&gt;F591</td>
</tr>
<tr>
<td>F58F</td>
<td>yes &gt;&gt;F59B</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;E9D9&gt;</td>
</tr>
<tr>
<td>F59B</td>
<td>copy file entry from directory &lt;E5AB&gt;</td>
</tr>
<tr>
<td>F59F</td>
<td>copy blocks used count to entry (D818)</td>
</tr>
<tr>
<td>F5AF</td>
<td>copy new EOF (D815)</td>
</tr>
<tr>
<td>F5A1</td>
<td>and new key block no. (D816)</td>
</tr>
<tr>
<td>F5C3</td>
<td>isolate new storage type (D805)</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;E8B6&gt;</td>
</tr>
<tr>
<td>F5DB</td>
<td>error? &gt;&gt;F5F7</td>
</tr>
<tr>
<td>F5E9</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>F5EE</td>
<td>yes, exit now &gt;&gt;F5F5</td>
</tr>
<tr>
<td>F5F0</td>
<td>yes, checkpoint it also &lt;EB9J&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 ********************

F5F7 | is this a close or flush all? |
F5FC | no >>F604 |
F600 | yes, save error code (E8BE) |
F603 | RETURN |

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

F606 | load FCB flags (FE92) |
F609 | from FCB (D88B) |
F60C | and exit |

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

F60D | exit with file access error code |
F618 | RETURN |
ProDOS MLI -- V1.1.1 -- 18 SEP 84

F71A copy new storage type (FECl)
F727 turn off all block allocation flags (EAP)
F72A update VCB free block count (F9F3)
F734 copy mark (Dh12)
F73C force current mark to infinity (Dh12)
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 (Dh15)
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 (F65F)
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

F7DB copy data to caller's paralist (F80C)
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 (FE8C)
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 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? >>F889
F855 copy new name to device's VCB (D700)
F861 exit, no errors
F862 RETURN
Beneath Apple ProDOS Supplement

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

---

*** RENAME FILE ***

F863 get path index <F959>
F866 copy old name with prefix to my buffer (D7)
F872 copy new name to buffer <F94B>
F875 error? >>F889
F877 get path index <F959>
F87D compare all levels of names up to and (DC0=)
F880 including the last. Find first which
F881 differ.
F885 save indicies into names which point to (F-
F888 final name. (FEB0)
F88B ---
F889 exit if they match completely
F896 RETURN

F897 index to differing new name (PEG0)
F89A point past it (D700)
F8A2 must be the lastl (D700)
F8A5 it isn't >>F887
F8A7 it is, (PEG0)
F8AA do the same with the old name (DC00)
F8B5 difference is only in last index? >>F88B
F8B7 no, bad path error
F8B9 ---
F8BA RETURN

F8BB names good. follow path to new file <E5B6=:
F8BE better get an error >>F8C4
F8C0 if found, duplicate name in directory
F8C3 RETURN

F8C4 if error, better be file not found
F8C6 or else its really an error... >>F889
F8CB copy old pathname again <E08A>
F8CB get its file entry <E5A3>
F8CB error? >>F8B9
F8D0 search FCB's <EFA3>
F8D5 exit if the file is open for write >>F889
F8DA does ACCESS permit rename?
F8DC yes >>F8BE
F8DE no, access error
F8E0 ---
F8E1 RETURN

F8E2 get type/len from entry (F5F)
F8E7 DIR file?
F8E9 yes, ok >>F8F3
F8EB seedling, sapling or tree?
F8ED yes, ok >>F8F3

---

F92A error? >>F8B9
F92F copy new name to DIR HDR (D700)
F934 and update directory's key block <F93C>
F937 error? >>F889
F939 go update directory entry and exit >>E46C

F93C ********* COPY PATH TO BUF & WRITE ************

F93C copy type/len and path to my buffer
F948 go write the block >>EB8A

F94B ********* POINT TO NEW NAME *********************

F94B $48/$49 -- > second pathname
F956 go copy it >>EU95

F959 ********* LOAD PATH INDEX *************************

F959 load pathname index
F960 (including prefix if any) (BF9A)
F963 ---
F965 RETURN

F966 *********************** MLI DESTROY CALL **************

F966 get file entry <E0A3>
F969 error? >>F9B5
F96B find FCB if any <EB3>
F96E FCB open? (F97)
F971 no >>F977
F973 yes, file open error
F976 RETURN

F977 no free blocks needed
F97F go compute VCB free block count <E973>
F982 ok? >>F9B9
F984 error, disk full?
F987 no, real error >>F8B5

---

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

---

ADDR DESCRIPTION/CONTENTS

F8EF else, compatibility error
F8F3 copy new path again <F94B>
F8F6 error? >>F8B9
F8FB get length of last name (FEB9)
F903 copy it and name to file entry buffer (D700)
F913 combine new len with type (D700)
F919 DIR file?
F91B no, go update entry and exit >>F939
F91D yes, (F78)
F927 read key block of this subdirectory <EB8E>
Beneath Apple ProDOS Supplement

ProDOS MIL -- V1.1.1 -- 18 SEP 04

F989 DESTROY enabled in ACCESS? <<EA74>>
F98E yes >>F995
F990 no, access error
F995 check status of device <<FP86>>
***
F99B error? >>F9B5
F99D point to key block <<FE7E>>
F9AC DIR file?
F9BD no >>F9B6
F9B2 yes, handle differently >>FA8E
F9B5 RETURN

*** DESTROY NON-DIRECTORY FILE ***
F9B6 set new storage type <<FC81>>
F9BD zero EOF mark <<FC81>>
F9C3 byte offset = <<2400>>
F9CB free all blocks in file <<FA78>>
F9CB error? >>F9B5
F9CD free key block of seeding <<FC86>>
***
F9DE error? >>F9B5
F9DB mark DIR entry free
F9DD decrement DIR file count <<FE51>>
F9DB checkpoint volume bitmap <<EB93>>
F9EB error? >>F9B5
F9ED update free block count in VCB <<F9>>
F9F0 and go update the directory >>EB45

*** FREE WHOLE INDEX BLOCKS AFTER EOF ***
(Free 8 subindex blocks each time the master index block is read since we must share its buffer)
FAA2 copy up to 8 non-zero index blocks
FAA4 numbers to (DC80)
FAA7 a handy table <<FECA>>
FAA8 ---
FAA9 if there weren't 8 left to do, zero (FECA)
FAA4 remainder of the table (FED2)
FAA9 ---
ProDOS ML1 -- Vi.1.1 -- 18 SEP 84

ADDR  DEC

PAC8  uP:CEPTION/CONTENTS
FAD0  fac
FAD3  s
FAD6  (D:st master index counter)
FAD8J  (C:all block) (FEC9) : (FEC8)
FAD9  e: 1KNUM (FECA)
FAE7  ftik entry when a 1 entry is for
FAE8  ed the sub-index block <3h1> >>FAA5
FAF0  d?? ??FAF5
FAF2  lls all blocks <FB86>
FAF4  m?? ??FAF5
FAF5  Rd: loop to do all <FA
FAF6  nW while exit >>FAA5
FAF9  WTMR
FAFA  e:
FAPF  x: go free all the sub-in
FB02  xtish follow EOF <FB08> d blocks (FC04)
FB04  d?? ??FAF5
FB07  i:fe back master index <1h1>
FB09  d?? ??FAF5
FB0C  o: in first subindex? (FC02)
FB0E  (D:go, demote to sapling FCB)
FB10  else, BLKNUM = subindex file >>FB8E
FB12  al:ains the EOF mark which (DC08)
FB1D  exit if none there) >>FA
FB14  C: read subindex block <EBBE>
FB1E  d: and continue below >>FB20
FB21  exist there is an error
skote tree to sapling <FB
FB23  r
FB26  e: *** TRUNCATE SAPLING FILE ***
FB28  g
FB2C  x:ed key block <FB87>
FB2E  d?? ??FAF5
FB31  ft: LSB of block number (1)
FB33  W: zero, no blocks to free >>FB38
FB36  else, free rest of blocks >>FB38
FB38  o:owing the EOF, check in index <FB08>
FB3B  x:te index block back <E: or error >>FAF5
FB3D  m?? ??FAF5
FB40  ft: LSB of block number (1)
FB45  (ight be block ? >>FB52 (FC05)
FB4C  r: get BLKNUM of data bl
FB4F  al: index block lock (DC06)
FB51  w: block allocated?) >>FA
FB54  d: data block <EBBE>
FB61  x: continue below >>FB61

ProDOS ML1 -- Vi.1.1 -- 18 SEP 84

ADDR  DESCRIPTION/CONTENTS

FB52  back to block Ø? (FC04)
FB55  m >>FB3D
FB57  s, demote to seedling <FB94>
FB5A  d?? ??FB6

*** TRUNCATE SEEDLING FILE ***

FB5C  read key block <FB87>
FB5F  d?? ??FB6
FB61  x:ed page? (FC07)
FB64  yes ??FB6C
FB67  no, better be second >>FB85
FB69  get byte offset (FC06)
FB6C  
FB6E  zero beyond EOF mark (DC08)
FB7C  in both pages if necessary (DC08)
FB82  then write block back and exit >>EBBE
FB85  exit normally
FB86  RETURN

FB87  *********** READ KEY BLOCK *************
FB87  BLKNUM = key block number (FE0B)
FB91  exit by reading the block >>EBBE

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

FB94  free block (FC09)
FB9D  d?? ??FB5
FB9F  get block from old index (DC08)
FBAC  reduce storage type by one (FC01)
FBAD  exit
FBBD  RETURN

FB86  *********** FREE ALL BLOCKS IN AN INDEX BLK ***********

FB66  
FB88  save BLKNUM
FBBE  for each index entry after mark, (FB9D)
FBC9  if it is non-zero...
FBDD  free the block <EA34>
FBD3  d?? ??FB64
FBD5  zero the index entry now (FE9D)
FBF0  
FBE1  loop through all entries >>FBBE
FBE4  
FB6E  restore old BLKNUM
FB6C  exit
**FREE I/O BUFFER**

FC4A is buffer already free? <FC3C>
FC4F yes, exit >>FC71
FC51 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

---

**LOCATE BIT MAP POSITION**

(GIVEN PAGE NUMBER)

FC73 XREG contains page number
FC74 compute page number times 8
FC77 use as offset for bitmask (BF80)
FC7E page number / 8 = byte offset
FC7F into bitmap
Beneath Apple ProDOS Supplement

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

ADDRESS DESCRIPTION/CONTENTS

FD05  Set Y = Ø
FD06  3 pages of code to copy
FD08  ---
FD09  copy quit code handler to $1000
FD17  Restore zero page to original state
FD23  enable HIGH RAM BANK1 (C0B8)
FD26  (MLI) (C0B8)
FD2B  point RESET vector at $1000 (03F2)
FD33  set power-up byte properly
FD38  go to quit code handler at $1000 >>1000

FD3B  ********** NEW ROUTINE **************
THE ADDRESS OF THIS ROUTINE IS AT $3EA.
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 (BF0F)
FD56  enable RAM, BANK1 (C0B8)
FD5C  restore original P-reg
FD5E  if error number is zero, (BF0F)
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:
ØØØØ EFGH
+ØØØØ ABCD

FD65  GET BUF
FD66  UNUSED
FD67  UNUSED
FD68  UNUSED
FD69  ALLOC INTERRUPT

FD6A  DEALLOC INTERRUPT
FD6B  UNUSED
FD6C  UNUSED
FD6D  READ BLOCK
FD6E  WRITE BLOCK
FD6F  GET TIME
FD70  EXIT
FD71  CREATE
FD72  DESTROY
FD73  RENAME
FD74  SET FILE INFO
FD75  GET FILE INFO
FD76  ON LINE
FD77  SET PREFIX
FD78  GET PREFIX
FD79  OPEN
FD7A  NEWLINE
FD7B  READ
FD7C  WRITE
FD7D  CLOSE
FD7E  FLUSH
FD7F  SET MARK
FD80  GET MARK
FD81  UNUSED
FD82  SET EOF
FD83  GET EGF
FD84  SET BUF

FD85  ********** PARAMETER COUNT TABLE **********

FD85  GET BUF
FD86  UNUSED
FD87  UNUSED
FD88  UNUSED
FD89  ALLOC INTERRUPT
FD8A  DEALLOC INTERRUPT
FD8B  UNUSED
FD8C  UNUSED
FD8D  READ BLOCK
FD8E  WRITE BLOCK
FD8F  GET TIME
FD90  EXIT
FD91  CREATE
FD92  DESTROY
FD93  RENAME
FD94  SET FILE INFO
FD95  GET FILE INFO
FD96  ON LINE
FD97  SET PREFIX
FD98  GET PREFIX
FD99  OPEN

NEXT OBJECT ADDR: FD05
ProDOS MLI -- V1.1.1 -- 18 SEP 84

ADDRESSES

FD9A NEWLINE
FD9B READ
FD9C WRITE
FD9D CLOSE
FD9E FLUSH
FD9F SET MARK
FDA0 GET MARK
FDA1 UNUSED
FDA2 SET EOF
FDA3 GET EOF
FDA4 SET BUF

FDA5 ********** MLI COMMAND ADDRESS TABLE ******************************

FD5A CREATE
FD5B DESTROY
FD5C RENAME
FD5D SET FILE INFO
FD5E GET FILE INFO
FD5F ON LINE
FD60 SET PREFIX
FD61 GET PREFIX
FD62 OPEN
FD63 NEWLINE
FD64 READ
FD65 WRITE
FD66 CLOSE
FD67 FLUSH
FD68 SET MARK
FD69 GET MARK
FD6A SET EOF
FD6B GET EOF
FD6C SET BUF
FD6D GET BUF

FDCC ********** MLI COMMAND INFO BYTE ******************************

| PATHNAME FLAG |
| REFERENCE NUMBER FLAG |
| DATETIME STAMP FLAG |
| COMMAND NUMBER |

FDCC 1 0 1  00
FDCE 1 0 1  01
FDCF 1 0 1  02
FDCH 1 0 1  03
FD9A 0 0 0  04
FD9B 0 0 0  05
FD9C 0 0 0  06
FD9D 0 0 0  07
FD9E 0 0 0  08

FE00 ********** BITMASK TABLE ******************************

FE00 10000000
FE01 01000000
FE02 00100000
FE03 00010000
FE04 00001000
FE05 00000100
FE06 00000010
FE07 00000001

FE00 ********** OFFSETS TO DATA AT $F300 ******************************
<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>FE0E</td>
<td>Key Block</td>
</tr>
<tr>
<td>FE0A</td>
<td># Blocks Used</td>
</tr>
<tr>
<td>FE0C</td>
<td>End of File</td>
</tr>
<tr>
<td>FE0F</td>
<td>SET/GET FILE_INFO OFFSETS</td>
</tr>
<tr>
<td>FE0F</td>
<td>Access</td>
</tr>
<tr>
<td>FE10</td>
<td>File Type</td>
</tr>
<tr>
<td>FE11</td>
<td>Aux Type</td>
</tr>
<tr>
<td>FE13</td>
<td>Storage Type</td>
</tr>
<tr>
<td>FE14</td>
<td>Blocks Used (MSB on means GET only no SET)</td>
</tr>
<tr>
<td>FE16</td>
<td>Datetime (Last Mod)</td>
</tr>
<tr>
<td>FE1A</td>
<td>Datetime (Creation)</td>
</tr>
<tr>
<td>FE1F</td>
<td>FATAL ERROR MESSAGE</td>
</tr>
<tr>
<td>FE1E</td>
<td>INSERT SYSTEM DISK AND RESTART</td>
</tr>
<tr>
<td>FE46</td>
<td>VARIABLES - DATA AREA</td>
</tr>
</tbody>
</table>

**Parent Pointer Block**

**Parent Entry Number**

**Parent Entry Length**

**Datetime (Creation)**

**Datetime (Last Mod)**

**Variables - Data Area**

**FE0E** 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)

**FE1A** Datetime (Creation)

**FE1F** FATAL ERROR MESSAGE

**FE1E** INSERT SYSTEM DISK AND RESTART

**FE46** VARIABLES - DATA AREA

**FE46** Parent Pointer Block

**FE48** Parent Entry Number

**FE49** Parent Entry Length

**FE4A** Datetime (Creation)

**FE4E** Version

**FE4F** Min Version

**FE50** Access Byte

**FE51** Entry Length

**FE52** Entries per Block

**FE53** File Count

**FE55** Bitmap Pointer

**FE57** Total Blocks

THE FOLLOWING 6 BYTES UNIQUELY IDENTIFY A FILE:

**FE59** Device Number

**FE5A** Current Directory Block Number (HDR)

**FE5C** Block Number of File Entry in Directory

**FE5E** File Entry Number in Directory

**FE5F** TYPE/LENGTH

**FE60** File Name

**FE66** File Type

**FE70** Key Pointer

**FE72** Blocks Used

**FE74** End of File

**FE77** Datetime (Last Mod)

**FE78** Version

**FE7C** Min Version

**FE7D** Access Attr

**FE7E** Aux Type

**FE80** Datetime (Creation)

**FE84** Header Point

**FE86** Variants - Data Area

**FE86** 3 Byte Scratch Block

**FE89** Entrys

**FE8A** End of File

**FE8D** Previous Map

**FE90** Compare Vol/Fldr

**FE91** Offset into bitmap

**FE92** Offset into bitmap

**FE93** Free FCN for bitmap

**FE94** Number of Blocks

**FE96** Storage Type/Size/Dirty Number of Offset

**FE97** FCN already Offset

**FE98** File Counter Page

**FE99** Entries/Bit

**FE9A** Free Entries # of Item

**FE9B** bit for free

**FE9C** # Blocks in FCN

**FE9D** Y Register

**FE9E** Pathname Len

**FE9F** Devnum for

**FEA0** Block of PM

**FEA2** Bitmap Byte

**FEA3** Bitmap Page

**FEA4** Bitmap Buff
DOS MLI -- V1.1.1 -- 18 SEP 84

NEXT OBJECT ADDR: FE65

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>EA5</td>
<td>Bitmap Flag (if $80, needs writing)</td>
</tr>
<tr>
<td>EA6</td>
<td>Bitmap DEVNUM</td>
</tr>
<tr>
<td>EA7</td>
<td>Bitmap Block Number</td>
</tr>
<tr>
<td>EA9</td>
<td>Bitmap Block offset for Multiblock Bitmaps</td>
</tr>
<tr>
<td>EAA</td>
<td>New Mark to be Positioned to for Set Mark</td>
</tr>
<tr>
<td>EAD</td>
<td>or New Moving Mark (for READ)</td>
</tr>
<tr>
<td>EAF</td>
<td>or New EOF for SET_EOF</td>
</tr>
<tr>
<td>EB0</td>
<td>Request Count (Read/Write etc.)</td>
</tr>
<tr>
<td>EB1</td>
<td>Multi-Block I/O count</td>
</tr>
<tr>
<td>EB2</td>
<td>Newline character</td>
</tr>
<tr>
<td>EB3</td>
<td>Newline mask</td>
</tr>
<tr>
<td>EB4</td>
<td>I/O Transfer occurred flag</td>
</tr>
<tr>
<td>EB3</td>
<td>MLI Command * 2</td>
</tr>
<tr>
<td>EB4</td>
<td>ORed into Access Flags ($20 - Backup)</td>
</tr>
<tr>
<td>EB5</td>
<td>Duplicate Volume Flag (if $FF)</td>
</tr>
<tr>
<td>EB6</td>
<td>Duplicate Volume's VCB index</td>
</tr>
<tr>
<td>EB7</td>
<td>MLI function code (low 5 bits)</td>
</tr>
<tr>
<td>EB8</td>
<td>Characters in current Pathname index lvl or</td>
</tr>
<tr>
<td>EB9</td>
<td>ONLINE: volname len - loop index</td>
</tr>
<tr>
<td>EB9</td>
<td>new pathname: index to last name</td>
</tr>
<tr>
<td>EBA</td>
<td>old pathname: index to last name or..</td>
</tr>
<tr>
<td>EBB</td>
<td>ONLINE: index to data buffer</td>
</tr>
<tr>
<td>EBC</td>
<td>Old PFIXPTR value</td>
</tr>
<tr>
<td>EBD</td>
<td>Pathname fully qualified flag (if $FF)</td>
</tr>
<tr>
<td>EBE</td>
<td>Pathname: temp save area for index or..</td>
</tr>
<tr>
<td>EBF</td>
<td>ONLINE: DEVCNT</td>
</tr>
<tr>
<td>ECF</td>
<td>close-all error code</td>
</tr>
<tr>
<td>ECF</td>
<td>Set EOF: new Key Block pointer</td>
</tr>
<tr>
<td>ECF</td>
<td>New storage type (SET_EOF)</td>
</tr>
<tr>
<td>ECF</td>
<td>Freed Blocks count</td>
</tr>
<tr>
<td>ECF</td>
<td>EOF Block number (MSB then LSB)</td>
</tr>
<tr>
<td>ECF</td>
<td>EOF byte offset into Block</td>
</tr>
<tr>
<td>ECF</td>
<td>EOF - Master index counter</td>
</tr>
<tr>
<td>ECF</td>
<td>Save area for index into table below</td>
</tr>
</tbody>
</table>

**DEVICE TABLE BUILT BY ONLINE**

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

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ECA</td>
<td>device table part one</td>
</tr>
<tr>
<td>EDA</td>
<td>device table part two</td>
</tr>
<tr>
<td>SDA</td>
<td>length of path, etc.</td>
</tr>
<tr>
<td>EDD</td>
<td>next buffer address</td>
</tr>
<tr>
<td>EDF</td>
<td>16 byte stack save area</td>
</tr>
<tr>
<td>EEF</td>
<td>6 byte zero page save area</td>
</tr>
<tr>
<td>EFP</td>
<td>Jump Vector, used for indirect jumps</td>
</tr>
</tbody>
</table>
ProDOS SYSTEM GLOBAL PAGE---MLI 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.
### ProDOS System Global Page

**NEXT OBJECT ADDRESS: BF80**

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

### General System Info

<table>
<thead>
<tr>
<th>ADDR</th>
<th>LABEL</th>
<th>CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>BF90-BF91</td>
<td>DATE</td>
<td>YYYYYMM MMMMDDD.</td>
</tr>
<tr>
<td>BF92-BF93</td>
<td>TIME</td>
<td>HHHHHH HHHHH.</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>
<tr>
<td></td>
<td></td>
<td>.00...0 = II</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.01...0 = II</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.10...0 = IIC</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.11...0 = IIII emulation</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.00...1 = Future expansion</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.01...1 = Future expansion</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.10...1 = Future expansion</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.11...1 = Future expansion</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.00... = Unused</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.01... = 48K</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.10... = 64K</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.11... = 128K</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.X... = Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.0...0 = No 80-column card</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.0...1 = 80-column card present</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.0...0 = No compatible clock present</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.0...1 = Card present</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Slot ROM map bit (on indicates ROM present).</td>
</tr>
<tr>
<td>BF99</td>
<td>SLTBYT</td>
<td>Prefix flag (0 indicates no active prefix).</td>
</tr>
<tr>
<td>BF9A</td>
<td>PREFIX</td>
<td>MLI active flag (1... indicates active).</td>
</tr>
<tr>
<td>BF9B</td>
<td>MLIACCTV</td>
<td>Last MLI call return address.</td>
</tr>
<tr>
<td>BF9C-BF9D</td>
<td>CMDADR</td>
<td>X-register savearea for MLI calls.</td>
</tr>
<tr>
<td>BF9E</td>
<td>SAVEX</td>
<td>Y-register savearea for MLI calls.</td>
</tr>
</tbody>
</table>

### Language Card Bank Switching Routines

**NEXT OBJECT ADDRESS: BF80**

<table>
<thead>
<tr>
<th>ADDR</th>
<th>LABEL</th>
<th>CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>BF80-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>YYYYYMM MMMMDDD.</td>
</tr>
<tr>
<td>BF92-BF93</td>
<td>TIME</td>
<td>HHHHHH HHHHH.</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>
<tr>
<td></td>
<td></td>
<td>.00...0 = II</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.01...0 = II</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.10...0 = IIC</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.11...0 = IIII emulation</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.00...1 = Future expansion</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.01...1 = Future expansion</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.10...1 = Future expansion</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.11...1 = Future expansion</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.00... = Unused</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.01... = 48K</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.10... = 64K</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.11... = 128K</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.X... = Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.0...0 = No 80-column card</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.0...1 = 80-column card present</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.0...0 = No compatible clock present</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.0...1 = Card present</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Slot ROM map bit (on indicates ROM present).</td>
</tr>
<tr>
<td>BF99</td>
<td>SLTBYT</td>
<td>Prefix flag (0 indicates no active prefix).</td>
</tr>
<tr>
<td>BF9A</td>
<td>PREFIX</td>
<td>MLI active flag (1... indicates active).</td>
</tr>
<tr>
<td>BF9B</td>
<td>MLIACCTV</td>
<td>Last MLI call return address.</td>
</tr>
<tr>
<td>BF9C-BF9D</td>
<td>CMDADR</td>
<td>X-register savearea for MLI calls.</td>
</tr>
<tr>
<td>BF9E</td>
<td>SAVEX</td>
<td>Y-register savearea for MLI calls.</td>
</tr>
</tbody>
</table>

### Interrupt Routines

**NEXT OBJECT ADDRESS: BF80**

<table>
<thead>
<tr>
<th>ADDR</th>
<th>LABEL</th>
<th>CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>BF80-BF8F</td>
<td>INTADDR</td>
<td>Interrupt return address.</td>
</tr>
</tbody>
</table>

### Language Card Bank Switching Routines

**NEXT OBJECT ADDRESS: BF80**

<table>
<thead>
<tr>
<th>ADDR</th>
<th>LABEL</th>
<th>CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>BF80-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>YYYYYMM MMMMDDD.</td>
</tr>
<tr>
<td>BF92-BF93</td>
<td>TIME</td>
<td>HHHHHH HHHHH.</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>
<tr>
<td></td>
<td></td>
<td>.00...0 = II</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.01...0 = II</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.10...0 = IIC</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.11...0 = IIII emulation</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.00...1 = Future expansion</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.01...1 = Future expansion</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.10...1 = Future expansion</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.11...1 = Future expansion</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.00... = Unused</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.01... = 48K</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.10... = 64K</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.11... = 128K</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.X... = Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.0...0 = No 80-column card</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.0...1 = 80-column card present</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.0...0 = No compatible clock present</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.0...1 = Card present</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Slot ROM map bit (on indicates ROM present).</td>
</tr>
<tr>
<td>BF99</td>
<td>SLTBYT</td>
<td>Prefix flag (0 indicates no active prefix).</td>
</tr>
<tr>
<td>BF9A</td>
<td>PREFIX</td>
<td>MLI active flag (1... indicates active).</td>
</tr>
<tr>
<td>BF9B</td>
<td>MLIACCTV</td>
<td>Last MLI call return address.</td>
</tr>
<tr>
<td>BF9C-BF9D</td>
<td>CMDADR</td>
<td>X-register savearea for MLI calls.</td>
</tr>
<tr>
<td>BF9E</td>
<td>SAVEX</td>
<td>Y-register savearea for MLI calls.</td>
</tr>
</tbody>
</table>

### Interrupt Routines

**NEXT OBJECT ADDRESS: BF80**

<table>
<thead>
<tr>
<th>ADDR</th>
<th>LABEL</th>
<th>CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>BF80-BF8F</td>
<td>INTADDR</td>
<td>Interrupt return address.</td>
</tr>
</tbody>
</table>

### Language Card Bank Switching Routines

**NEXT OBJECT ADDRESS: BF80**

<table>
<thead>
<tr>
<th>ADDR</th>
<th>LABEL</th>
<th>CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>BF80-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>YYYYYMM MMMMDDD.</td>
</tr>
<tr>
<td>BF92-BF93</td>
<td>TIME</td>
<td>HHHHHH HHHHH.</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>
<tr>
<td></td>
<td></td>
<td>.00...0 = II</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.01...0 = II</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.10...0 = IIC</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.11...0 = IIII emulation</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.00...1 = Future expansion</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.01...1 = Future expansion</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.10...1 = Future expansion</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.11...1 = Future expansion</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.00... = Unused</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.01... = 48K</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.10... = 64K</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.11... = 128K</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.X... = Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.0...0 = No 80-column card</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.0...1 = 80-column card present</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.0...0 = No compatible clock present</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.0...1 = Card present</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Slot ROM map bit (on indicates ROM present).</td>
</tr>
<tr>
<td>BF99</td>
<td>SLTBYT</td>
<td>Prefix flag (0 indicates no active prefix).</td>
</tr>
<tr>
<td>BF9A</td>
<td>PREFIX</td>
<td>MLI active flag (1... indicates active).</td>
</tr>
<tr>
<td>BF9B</td>
<td>MLIACCTV</td>
<td>Last MLI call return address.</td>
</tr>
<tr>
<td>BF9C-BF9D</td>
<td>CMDADR</td>
<td>X-register savearea for MLI calls.</td>
</tr>
<tr>
<td>BF9E</td>
<td>SAVEX</td>
<td>Y-register savearea for MLI calls.</td>
</tr>
</tbody>
</table>
Beneath Apple ProDOS Sc

ProDOS QUIT Code -- V1.1.1 -- 18 SEP 84

**CONTENTS**

**ADDR**  **DESCRIPTION/new ADDRESS**

1000  **MODULE START**

1000  ************ INITIALIZE **********

1000  Select ROM (C002)  (C00B)
1001  Set Video (P039)
1006  Set Keyboard (P099)
1009  Disable 80 column card (C003)
100C  Select Alternate character set
100F  Disable 80 column store (C000)

1012  ************ INITIALIZE MEMORY BITMA**********

1012  Mark pages $0, $1, $4 through $5B
1014  and $BF as in use

1027  ************ DISPLAY CURRENT PREFIX 11E9**

1027  Clear Screen and Home cursor<br>102A  Go down 1 line (FDCE)
102D  Get Pointer to Prompt1 (Prefix)
102F  and store it in Print Routine 1037  Call Print Routine (11E6)
103A  Position to line 3 1041  Call MLI (GET PREFIX) (BE00)
1044  Data: GET_PREFIX command number (1E9)
1045  Data: Pointer to Parameter list 1047  Terminate Prefix with 0 (0000)
104A  for Print routine 104F  Get Pointer to Prefix
1051  and store it in Print Routine 1059  And Print it (11E6)

185C  **GET PREFIX NAME *******

185C  Initialize counter 1063  Read a key (FD0C)
1066  Is it CARRIAGE RETURN? 1068  Yes, then accept Prefix >>1088 >>1027
106A  NO, then save character 106B  Clear to end of line (FC9C) >>1027
106D  Retrieve character 106F  Is it ESCAPE?
1071  Yes, then start all over aga
1073  Is it CANCEL?
1075  Yes, then start all over again 1077  Is it TAB?
1079  Yes, then sound Bell, get ano
107B  Is it BACKSPACE?
107D  NO, then keep checking >>108C
107F  YES, then is there room to move 1081  NO, then don't try >>1086
<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;1083</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;1083</td>
</tr>
<tr>
<td>1094</td>
<td>Is it less than or equal to &quot;Z&quot;?</td>
</tr>
<tr>
<td>1096</td>
<td>Yes, keep checking &gt;&gt;109A</td>
</tr>
<tr>
<td>1098</td>
<td>Turn off lowercase</td>
</tr>
<tr>
<td>109A</td>
<td>Is it less than &quot;,&quot;?</td>
</tr>
<tr>
<td>109C</td>
<td>Yes, Invalid - try again &gt;&gt;108E</td>
</tr>
<tr>
<td>109E</td>
<td>Is it greater than &quot;Z&quot;?</td>
</tr>
<tr>
<td>10AB</td>
<td>Yes, Invalid - try again &gt;&gt;108E</td>
</tr>
<tr>
<td>10A2</td>
<td>Is it less than or equal to &quot;+&quot;?</td>
</tr>
<tr>
<td>10A4</td>
<td>Yes, keep checking &gt;&gt;10AA</td>
</tr>
<tr>
<td>10A6</td>
<td>Is it less than &quot;A&quot;?</td>
</tr>
<tr>
<td>10A8</td>
<td>Yes, Invalid - try again &gt;&gt;108E</td>
</tr>
<tr>
<td>10AA</td>
<td>Else, valid character - increment counter</td>
</tr>
<tr>
<td>10AB</td>
<td>Found 39 characters</td>
</tr>
<tr>
<td>10AD</td>
<td>Yes, then start all over &gt;&gt;1075</td>
</tr>
<tr>
<td>10AF</td>
<td>Put valid character in buffer (0280)</td>
</tr>
<tr>
<td>10B2</td>
<td>And print it &lt;FDOE&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 ML1 (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>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>10CE</td>
<td>Clear Screen and Home cursor &lt;FC58&gt;</td>
</tr>
<tr>
<td>10D1</td>
<td>Go down 1 line &lt;FDE8&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>10E4</td>
<td>Output a RUB</td>
</tr>
<tr>
<td>10F1</td>
<td>Poll Keyboard latch (C000)</td>
</tr>
<tr>
<td>10F4</td>
<td>Loop until keypress found &gt;&gt;10F1</td>
</tr>
<tr>
<td>10F6</td>
<td>Clear latch (C810)</td>
</tr>
</tbody>
</table>

**10F9 ** **Is it ESCAPE? ** **-----------------------------------------------**

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<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;?</td>
</tr>
<tr>
<td>1126</td>
<td>Set character - try again &gt;&gt;1114</td>
</tr>
<tr>
<td>1128</td>
<td>Is it greater than &quot;Z&quot;?</td>
</tr>
<tr>
<td>112A</td>
<td>Yes, Invalid - try again &gt;&gt;1114</td>
</tr>
<tr>
<td>112C</td>
<td>Is it less than or equal to &quot;9&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;A&quot;?</td>
</tr>
<tr>
<td>1132</td>
<td>Yes, Invalid - try again &gt;&gt;1114</td>
</tr>
<tr>
<td>1134</td>
<td>Else, valid character - save it</td>
</tr>
<tr>
<td>1135</td>
<td>Clear to end of line &lt;FC9C&gt;</td>
</tr>
<tr>
<td>113B</td>
<td>Retrieve character</td>
</tr>
<tr>
<td>113D</td>
<td>Print it &lt;FDOE&gt;</td>
</tr>
<tr>
<td>113C</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>

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1147</td>
<td>Output a blank</td>
</tr>
<tr>
<td>114C</td>
<td>Store length of Application name (0280)</td>
</tr>
<tr>
<td>114F</td>
<td>Call ML1 (GET_FILE_INFO) &lt;BF00&gt;</td>
</tr>
<tr>
<td>1152</td>
<td>Data: GET_FILE_INFO command number</td>
</tr>
<tr>
<td>1153</td>
<td>Data: Pointer to Parameter list</td>
</tr>
<tr>
<td>1155</td>
<td>Continue if no error &gt;&gt;115A</td>
</tr>
<tr>
<td>1157</td>
<td>Else, go to Error Handler &gt;&gt;11F6</td>
</tr>
</tbody>
</table>
115A Get File Type (12D5)
115F No, indicate Error $01
1163 Go to Error Handler >>11F6
1166 Set Reference number to 0
116B Call MLI (CLOSE) <BF00>
116E Data: CLOSE command number
116F Data: Pointer to Parameter list
1171 Continue if no error >>1176
1173 Else, go to Error Handler >>11F6
1176 Get Access Byte (12D4)
1177 Yes, >>11B2
117D No, Indicate Error $27
117F Go to Error Handler >>11F6
1182 Call MLI (OPEN) <BF00>
1185 Data: OPEN command number
1186 Data: Pointer to Parameter list
1188 Continue if no error >>118D
118A Else, go to Error Handler >>11F6
118D Get Reference Number (12E8)
1190 and update READ and (12E6)
1193 GET_EOF parameters list (12F4)
1196 Call MLI (GET_EOF) <BF00>
1199 Data: GET_EOF command number
119A Data: Pointer to Parameter list
119C Continue if no error >>11A1
119E Else, go to Error Handler >>11F6
11A1 Is EOF mark less than $10000 (12F7)
11A4 Yes, continue on >>11A6
11A6 No, Indicate Error $27
11A8 Go to Error Handler >>11F6
11A9 Transfer EOF to Request count (12F5)
11A9 in READ parameter list (12EF)
11B7 Call MLI (READ) <BF00>
11B9 Data: READ command number
11BD Save status of READ
11BE Call MLI (CLOSE) <BF00>
11C1 Data: Get Prefix command number
11C2 Data: Pointer to Parameter list
11C6 Continue if no error >>11CA
11C7 Else, retrieve status
11CA and go to Error Handler >>11F6

---

11D0 ********* BACKSPACE ROUTINE ***********
11D6 Get cursor position horizontal
11D8 If 0 exit routine >>11E3
11D9 Decrement counter
11DA Output a space
11DB Move cursor back 2 spaces
11DE Output a space <FDED>
11E0 Move cursor back 1 space
11E2 Return to get another character >>10EA

---

11E6 ********* PRINT TEXT ROUTINE ***********
11E6 Initialize offset
11E7 Get a character (11E8)
11E9 If it is 0 then exit >>11F5
11EB Output it <FDED>
11EC Increment offset
11E3 Get another character unless we've done 256 >>11E8
11E5 Return to caller

---

11F6 ********* PRINT ERROR MESSAGE ***********
11F6 Save Accumulator (Error Number)
11F7 Position to line 12
11FF Get Error number
1201 Is it $01?
1203 No, then keep checking >>1211
1206 Get Pointer to Error1 (Not System file)
1207 and store it in Print Routine (11E9)
1209 Branch always taken >>1237
1211 Is it $40?
1213 Yes, then indicate Error3 >>1220
1215 Is it $46?
1217 Yes, then indicate Error3 >>1220
1219 Is it $45?
1221 Yes, then indicate Error3 >>1220
1223 Is it $46?
1225 Yes, then indicate Error3 >>1220
1227 Else, Get Pointer to Error2 (I/O Error)
1229 and store it 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)
1231 Print Error message <11E6>
1233 Position to line 0
**ProDOS QUIT Code -- V1.1.1 -- 18 SEP 84**

**NEXT OBJECT ADDR: 123E**

**DESCRIPTION/CONTENTS**

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'

12B1 **************************** PARAMETER LISTS ****************************

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

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

    CLOSE Parmlist
12E9 Parmcount
12EA Reference Number

ProDOS QUIT Code -- V1.1.1 -- 18 SEP 84

**NEXT OBJECT ADDR: 12E9**

**DESCRIPTION/CONTENTS**

    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 ****************************
Disk II Device Driver -- V1.1.1

**ADDRESS/DESCRIPTION CONTENTS**

**D000** MODULE STARTING ADDRESS

- 10 SEP 84 NEXT OBJECT ADDR: D000
- 5.25" DISK DRIVE
- RESIDES AT $D000
- VERSION 1.1.1

**D000** ********** ZERO PAGE EQUATIONS

- $D000-D06FF

- $D03A Checksum
- $D03A Workbyte
- $D03E Slot (Temporary) 10 SEP 84
- $D042 Command
- $D043 Unit Number
- $D044 I/O Buffer Pointer (low)
- $D045 I/O Buffer Pointer (high)
- $D046 Block Number (low)
- $D047 Block Number (high)

**D000** ********** INTERNAL EQUATIONS

- $D100 Dummy Block Buffer (1st)
- $D101 Dummy Block Buffer (2nd)

**D000** ********** EXTERNAL EQUATIONS

- $D200 Phase Zero Off
- $D208 Motor Off
- $D209 Motor On
- $D208 Drive Select half
- $D208 Read Data Register half
- $D208 Set Read Mode
d208 Set write mode
- $D208 Read Data Register (slot

**D000** ********** 5.25" DISK DRIVE

- $D200 Clear decimal mode
- $D201 Clear phases in case INT
- $D204 Five NOP's so code below
- $D205 fit up against Table at
- $D209 Check validity of caller
- $D20C If not valid exit with
- $D20E Convert Block Number to

- device in this slot $D60E
- will
- $D196
- 4 parameters $D6D8
- error $D34
- a Track and Sector

**D010** ********** PARAMETERS

- $D014 $D000-$0000
- $D015 Error Number
- $D017 $D03E
- $D018
- $D01A $D030
- $D01C
- $D01C Pointer
- $D01D
- $D01E
- $D01F
- $D020
- $D021
- $D022
- $D023
- $D024
- $D025
- $D026
- $D027
- $D028
- $D029
- $D030
- $D031
- $D032
- $D033
- $D034
- $D035
- $D036
- $D037
- $D038
- $D039
- $D03A
- $D03B
- $D03C
- $D03D
- $D03E
- $D03F
- $D040
- $D041
- $D042
- $D043
- $D044
- $D045
- $D046
- $D047
- $D048
- $D049
- $D04A
- $D04B
- $D04C
- $D04D
- $D04E
- $D04F
- $D050
- $D051
- $D052
- $D053
- $D054
- $D055
- $D056
- $D057
- $D058
- $D059
- $D05A
- $D05B
- $D05C
- $D05D
- $D05E
- $D05F
- $D060
- $D061
- $D062
- $D063
- $D064
- $D065
- $D066
- $D067
- $D068
- $D069
- $D06A
- $D06B
- $D06C
- $D06D
- $D06E
- $D06F
- $D070
- $D071
- $D072
- $D073
- $D074
- $D075
- $D076
- $D077
- $D078
- $D079
- $D07A
- $D07B
- $D07C
- $D07D
- $D07E
- $D07F
- $D080
- $D081
- $D082
- $D083
- $D084
- $D085
- $D086
- $D087
- $D088
- $D089
- $D08A
- $D08B
- $D08C
- $D08D
- $D08E
- $D08F
- $D090
- $D091
- $D092
- $D093
- $D094
- $D095
- $D096
- $D097
- $D098
- $D099
- $D09A
- $D09B
- $D09C
- $D09D
- $D09E
- $D09F
- $D0A0
- $D0A1
- $D0A2
- $D0A3
- $D0A4
- $D0A5
- $D0A6
- $D0A7
- $D0A8
- $D0A9
- $D0AA
- $D0AB
- $D0AC
- $D0AD
- $D0AE
- $D0AF
- $D0B0
- $D0B1
- $D0B2
- $D0B3
- $D0B4
- $D0B5
- $D0B6
- $D0B7
- $D0B8
- $D0B9
- $D0BA
- $D0BB
- $D0BC
- $D0BD
- $D0BE
- $D0BF
- $D0C0
- $D0C1
- $D0C2
- $D0C3
- $D0C4
- $D0C5
- $D0C6
- $D0C7
- $D0C8
- $D0C9
- $D0CA
- $D0CB
- $D0CC
- $D0CD
- $D0CE
- $D0CF
- $D0D0
- $D0D1
- $D0D2
- $D0D3
- $D0D4
- $D0D5
- $D0D6
- $D0D7
- $D0D8
- $D0D9
- $D0DA
- $D0DB
- $D0DC
- $D0DD
- $D0DE
- $D0DF
- $D0E0
- $D0E1
- $D0E2
- $D0E3
- $D0E4
- $D0E5
- $D0E6
- $D0E7
- $D0E8
- $D0E9
- $D0EA
- $D0EB
- $D0EC
- $D0ED
- $D0EE
- $D0EF
- $D0F0
- $D0F1
- $D0F2
- $D0F3
- $D0F4
- $D0F5
- $D0F6
- $D0F7
- $D0F8
- $D0F9
- $D0FA
- $D0FB
- $D0FC
- $D0FD
- $D0FE
- $D0FF
Drive to speed <D3B5>
on yet? <D4A0>
exit with error >>D0EA
and a "status" request?
don determine status >>D0FD
and a "read" request?

In continue on >>D09B
for data for write (pribilize) <D5F0>

Size "retry" count at 64 (D369)

D07F Wait on address field - Good read? <D39B>
D081 coma. keep continue on >>D0BE
D089 Is "not retry" count - More to try? (D369)
D08C No, then try again >>D09D
D091 Yes, keep in case indicate "I/O Error"
D090 Yes, keep "recalibration" count - More to try? (D36A)
D092 Is con exit with error >>D0EA
D093 Yes, "current" track (D35A)
D095 Prepare it
D09B --- Hit and
D09A Initialize it for recalibration
D09D --- Blize Retry Count
D09F Read a always taken >>D0CC
D0A2 Yes, if right track found? (D35A)
D0A4 Dece res on continue on >>D0D5
D0A7 Yes, "current" track (D35A)
D0A9 No, jus it
D0AB Dece res we found
D0AE No, bu it
D0B0 Get the value in Device Track Table <D4D3>
D0B3 Pres ask we want
D0B4 Double there <D1B6>
D0B5 add a always taken >>D09D
D0B7 Reinit ri sector found? (D357)
D0BC Breciif try again >>D0A4
D0C1 Was n't a "write" request?
D0C4 Yes, then go do it >>D0F4
D0C6 Get the data - Good read? <D3FD>
D0C9 Prese r try again >>D0A4
D0CA Get the no errors
D0CB Double structr, never taken
D0CC Put the error
D0CF Get the error number (D35B)
D0D0 And go it
D0D3 Brach tor off (C088)
D0DB Was th to caller
D0DB No, the
D0DF Is con
D0E0 Yes, t
D0E2 Read it
D0E5 No, t
D0E7 Indices
D0E9 BNE l
D0EA Indio
D0EB Prere
D0E0 Get S
D0F0 Turn c
D0F3 Return
Disk II Device Driver -- V1.1.1 -- 18 SEP 84  NEXT OBJECT ADDR: D1C4

ADDR  DESCRIPTION/CONTENTS

D200     ********** TABLE 2 ***************************************

Read Translate

D200     Entry for Bit Mask 1
D200     Entry for Bit Mask 2
D200     Entry for Bit Mask 3
D200     Entry for Write Translate

D300     ********** AUXILIARY BUFFER ******************************

D300     Auxiliary Buffer ($56 bytes) >>0056

D356     ********** VARIABLE AREA *******************************

D356     Track number
D357     Sector number
D358     Error number

****** TABLE 1 ***********************************************

Write Translate Table

Every 4th byte starting at $D203

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

D200     Entry for Bit Mask 1
D200     Entry for Bit Mask 2
D200     Entry for Bit Mask 3
D200     Entry for Write Translate

D300     ********** AUXILIARY BUFFER ******************************

D300     Auxiliary Buffer ($56 bytes) >>0056

D356     ********** VARIABLE AREA *******************************

D356     Track number
D357     Sector number
D358     Error number

****** TABLE 2 ***********************************************

Read Translate

D200     ********** TABLE 2 ***************************************

Write Translate Table

Every 4th byte starting at $D203

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

D200     Entry for Bit Mask 1
D200     Entry for Bit Mask 2
D200     Entry for Bit Mask 3
D200     Entry for Write Translate

D300     ********** AUXILIARY BUFFER ******************************

D300     Auxiliary Buffer ($56 bytes) >>0056

D356     ********** VARIABLE AREA *******************************

D356     Track number
D357     Sector number
D358     Error number

****** TABLE 1 ***********************************************

Read Translate Table with PreNiblize

Bit mask Tables and Epilog Table in unused areas

Read Translate

Bit Mask 1

00000000
10000000
NEXT OBJECT ADDR: D3C4

X (C08C)
X1
X (C08C)
XX (D36B)
F, Checksum (D36D)

By et?

FB
FB
FB

UUTINE ******************

X (C3)
Letter (D45A)

Address (D4AF)
Letter (D4B0)

$54 (D497)
Buffer (D498)

$AB (D470)
Buffer (D471)
Beneath Apple ProDOS Supplement

Disk II Device Driver -- V1.1.1 -- 18 SEP 84
NEXT OBJECT ADDR: D43

ADDRESS/DESCRIPTION/CONTENTS

D43
Loop until data valid >>D40

D44
Is is 2nd header mark ($9A)?

D44
No, then see if it is last header mark >>D43B

D44
Delay for register to clear

D44A
Read data register (C98C)

D44D
Loop until data valid >>D44A

D44F
Is it 3rd header mark ($6A)?

D451
No, then see if it is last header mark >>D43B

D453
Initialize offset into data buffer

D455
Initialize checksum

D459
Read a data byte (C98C)

D45E
Translate it (D100)

D461
Store it in Auxiliary buffer (D256)

D464
Compute running checksum

D466
Increment offset - More to do?

D467
Yes, then continue >>D457

D469
Reinitialize offset into data buffer

D46B
Branch always taken >>D472

D46D
Set carry flag indicating error

D46E
Return to caller

D46F
Store byte in Primary buffer (bottom third) (1000)

D472
Read a data byte (C98C)

D477
Translate it and merge in (D100)

D47A
bits from Auxiliary buffer (D256)

D480
Increment offset - done yet?

D481
No, then do another >>D46F

D483
Save last byte for later, no time now

D484
Strip off last two bits XXXXXX0

D486
Reinitialize offset

D48B
Read a byte (C98C)

D48D
Translate it and merge in (D100)

D490
bits from Auxiliary buffer (D256)

D496
Store byte in Primary buffer (middle third) (1000)

D499
Increment offset - done yet?

D49A
No, then do another >>D480

D49C
Read a byte (C98C)

D4A1
Strip off last two bits XXXXXX0

D4A3
Reinitialize offset

D4A5
Translate byte and merge in (D100)

D4AB
bits from Auxiliary buffer (D254)

D4AE
Store byte in Primary buffer (top third) (1000)

D4B1
Read a byte (C98C)

D4B6
Increment offset - done yet?

D4B7
No, then do another >>D4A5

D4B9
Strip off last two bits XXXXXX0

D4BB
Is checksum valid? (D100)

D4BE
No, then exit with error >>D4CC

D4C0
Get slot number

D4C2
Read data register (C88C)

D4C5
Loop until data valid >>D4C2

D4C7
Is is last trailing mark ($DE)?

Disk II Device Driver -- V1.1.1 -- 18 SEP 84
NEXT OBJECT ADDR: D43

ADDRESS/DESCRIPTION/CONTENTS

D4C9
Yes, then continue with carry clear >>D4CD

D4CC
Set Carry flag indicating error

D4CD
Get byte we stored away, we have time now

D4CE
Set proper offset

D4D0
Store byte in Primary buffer (offset $55)

D4D2
Return to caller

D4D3
******** UPDATE DEVICE TRACK TABLE ***************

D4D3
Get offset into Device Track Table (D4F1)

D4D6
Update Device Track Table (D359)

D4D9
Return to caller

D4DA
******** DETERMINE IF DRIVE IS ON (DATA CHANGING)********

D4DA
Get slot number

D4DC
Initialize counter

D4DE
Read data register (C98C)

D4EA
Delay 25 cycles <<D4F0

D4E9
Has data register changed? (C98C)

D4EB
Yes, then exit >>D4F0

D4ED
Just in case indicate No Device Connected Error

D4EE
No, try again >>D4DE

D4F0
Return to caller

D4F1
******** CONVET SLOT/DRIIVE TO TABLE OFFSET **********

D4F1
Preserve A-register

D4F2
Get Unit number DSSS0000

D4F4
Divide by 16 0000DSSS

D4F5
Put Drive into Carry 0000DSSS D

D4FA
Strip out Drive 0000DSSS D

D4FC
Roll left 0000S0SD

D4FD
Put result in X-register

D4FE
Restore A-register

D4FF
Return to caller

D500
******** WRITE DATA ROUTINE ***************

D500
Set Carry flag (anticipate error)

D504
Is diskette "write-protected"? (C98E)

D507
No, then continue on >>D50C

D509
Go to error routine >>D50F

D50C
Put transition byte from secondary buffer (D308)

D50F
Into zero page for timing

D511
Use SP for "sync" byte

D513
Write first "sync" byte (C98F)

D519
Set counter for four more

D51C
Delay so that writes occur

D51D
Exactly on 40 cycle loops
Disk II Device Driver -- V1.1.1 -- 18 SEP 84  NEXT OBJECT ADDR: D51E

ADDR    DESCRIPTION/CONTENTS

D51E  ** Sync byte <D5E> ?
D520  Write "st counter, done yes
D523  Decrement do another >>D51F)
D524  No, the next data mark <$2A>
D526  Write second data mark<$A>
D52B  Write third data mark <$A>
D530  Write the checksum auxiliary buffer
D535  Initialize index into Aux
D536  Index always taken >>D52)
D538  Branch a byte (Auxiliary >> buffer) (D200)
D53A  Get data or with previous lookup
D53D  Exclusive last two bits for X-reg for the (D203)
D540  Put residisk byte in tab1
D541  Lookup
D544  Get a disk byte (C08D) in Auxiliary buffer?
D546  Write "it index - Done with
D54C  Decrement another byte >> buffer
D54D  No, the next byte of Auxiliary recovery buffer
D54F  Get last the index into primary byte (1000)
D551  Initialize or with next FXXXX00
D553  Exclusive last two bits for cache lookup
D556  Strip them in X-reg for the (D203)
D558  Put residisk byte in tab1
D559  Lookup
D55C  Get a disk byte (C08D)  (1000)
D55E  Write "a byte (Primary buffer page"
D564  Get data offset, end 02F3
D567  Increment continue on >> P? boundary?
D569  No, the start page on page 2m >> D599
D56A  Did buffer go write checks page boundary?
D56C  Yes, then start one past >> D5B3
D56E  Did buffer go write last by buffer end
D570  Yes, the indicates odd or even
D572  Carry 1 st iteration byte
D573  Get trak (C08D)
D575  Write 2nd transition byte timing
D57B  Get new cycles for correct on odd byte?
D57D  Delay 2nd offset, buffer then >>D599
D57E  Increment we if we're done a byte (1100)
D57F  Yes, go or with next data FXXXX00
D581  Exclusive last two bits for cache lookup
D584  Strip bit in X-reg for the (D203)
D586  Put residisk byte in tab1
D588  Get a disk byte (C08D) (1000)
D58C  Write "a byte (Primary buffer"
D592  Get data offset a byte (1100)
D595  Increment or with next data in carry
D596  Exclusive buffer? - Put result FXXXX00
D599  End of X-reg last two bits
D59B  Strip on 1

Disk II Device Driver -- V1.1.1 -- 18 SEP 84  NEXT OBJECT ADDR: D59D

ADDR    DESCRIPTION/CONTENTS

D59D  Put result in X-reg for table lookup
D59E  Lookup "disk byte" in table (D203)
D5A1  Get slot
D5A3  Write "disk byte" (C08D)
D5A9  Get data byte (Primary buffer - page 2) (1100)
D5AC  Increment offset - Done yet?
D5AD  No, then do another >>D5B1
D5AF  Yes, then go write checksum >>D5B1
D5B1  --- >> D5C0
D5B3  Get last byte
D5B6  Write it (C08D)
D5B8  Delay 14 cycles for correct timing
D5C0  Use last byte in Primary buffer as checksum
D5C2  Lookup "disk byte" (D203)
D5C5  Get slot
D5C7  Write "disk byte" (C08D)
D5CD  Initialize offset into "epilog" table
D5CF  Delay 11 cycles for correct timing
D5D3  Load "epilog" from table ($DE, $AA, $EB, $FF) (D1C4)
D5D6  Go write it (D5B9)
D5D9  Increment offset
D5DA  Done all four yet?
D5DB  No, then do another >>D5D3
D5DE  Clear Carry flag (no error)
D5DF  Select read mode (C08B)
D5E5  Return to caller

D5E6  ********** WRITE A BYTE SUBROUTINE *****************************************

D5E6  Wait 9 cycles before write
D5E7  Wait 7 cycles before write
D5E9  Put A-register in data register (C08D)
D5EC  And write data register (C08C)
D5EF  Return to caller

D5F0  *********** PREINITIZE 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) FXXXX00 (1000)
D61D  Get last two bits 0000AB
D61F  Put in X-reg for table lookup
D628 Use lookup
D629 Get current
D632 Get last two
D633 Merge in m
D639 Save result (middle third)
D63A Get offset
D63B Compute offset for table lookup
D63D Put in X-re: value from stack
D63E Get data by word using table
D63F Store it in on stack
D642 Increment value (top third)
D643 No, then put bits
D645 Get low ord. for table lookup
D647 Subtract 1...value of stack
D648 Save it for bits using table
D649 Get low ord. on stack
D64C Modify code in primary buffer
D64F Buffer on fast into auxiliary buffer
D651 Else, compute
D654 Get byte (Auxiliary buffer (D388)
D656 Point at offset
D657 Exclusive-a another >>D61A
D659 Strip off last byte
D65B Put in X-re (offset in case there is (C80))
D65C Get "disk 1: last"
D65F Save result byte of buffer
D661 Buffer on in Write Data Routine (offset) (D552)
D663 Get offset-page boundary? - Yes, skip ahead >>D65F
D665 Carry indicate offset to last byte
D666 Get byte (boundary
D668 Did buffer age (boundary -1)
D66A Point at next byte (page boundary)
D66B Exclusive-if the together XXXXXXXX
D66D Save result last two bits XXXX0000
D66F Point at last for table lookup
D671 Get last byte from table (transition byte) (D283)
D673 Strip off (0 indicates page boundary)
D675 Save result page boundary? - Yes, skip ahead >>D66F
D677 Get high ord. last byte in buffer
D679 Modify codes (DD or even buffer start
D68C Get slot page (boundary)
D68E Modify code in Write Data Routine (D55D)
D69A Return to caller
D69B Determine if slot/drive has changed
D69E Put "current" drive in Carry
D69F Has slot changed? - No, then exit >>D6BD
D69G Get "current" slot
D69H Put in X-register
D69I Exit if Slot 0 >>D68D
D69J Is "current" motor is on? <D4C>
D69K No, then exit >>D6BD
D69L Wait until "current" motor is off (D370)
D69M Or else timeout >>D6A6
D69N Return to caller
D69E Clear IOM phases
D6CE Get unit number
D6C6 Strip drive bit
D6C8 Put slot 8 in X-Register
D6C9 Get Block Number
D6CF Return to caller
D6D0 Check calling parameters
D6D2 Is it greater or equal to 7?
D6D4 No, if greater than or equal to $11B >>D68D
D6D5 Indicate error
D6D7 Return to caller
D6D8 All is well
D6D9 Return to caller
D6EA Not used
IRQ Handler -- V1.1.1 -- 18 SEP 84

**ADDRESS 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)

**GLOBAL PAGE EQUATES**

- $BF56: Temporary storage 1
- $BF57: Temporary storage 2
- $BF88: A register savearea
- $BF8D: Bank ID byte
- $BF3D: IRQ exit code

**EXTERNAL EQUATES**

- $D000: RAM/ROM test byte
- $C082: ROM Select
- $C08B: BANK1 Select

**IRQ CODE**

- FF9b: Put A-Register on stack
- FF9C: Get Accumulator value from $45
- FF9E: and save it ($BF56)
- FF91: Replace $45 with A-Register
- FF92: since it may have been destroyed
- FF94: Load Status register
- FF95: Restore onto stack
- FF96: Isolate B flag - Was it a BRK?
- FF98: Yes, skip Interrupt stuff >>FFC2
- FF9A: Else, Check location $D000 ($D000)
- FF9D: Do we have RAM active
- FF9F: Yes, indicate so >>FFB3
- FF91: Else, indicate ROM
- FF93: Update Bank ID byte ($BF8D)
- FF96: Also save temporarily ($BF57)
- FF99: Push ($BF50) address of
- FF9B: routine to bank in Ram and
- FFBC: call IRQ on the stack
- FFBB: Push a new P-Register on stack with
- FF21: the Interrupt Disable flag set
- FF22: Push ($FA41) address less 1 of
- FF24: Monitor IRQ on the stack

**FFC8**

- Select ROM - execution continues in ROM ($C082)

**RESET CODE**

- FFCA: Push ($FA61) address less 1 of (FFD7)
- FFCE: Hardware Reset routine on to stack
- FFDD: Exit via select ROM code above >>FFC8
- FFDB: Address (-1) of Hardware Reset routine

**IRQ CODE**

- Called via $BF50 in System Global Page

- FFCA: Save Accumulator in Global page ($BF88)
- FFDB: Restore $45 with original value ($BF56)
- FFDE: Select RAM (read & write) ($C088)
- FF9E: use BANK1 ($C08B)
- FF96: Get Bank ID byte ($BF57)
- FF99: Leave via Global Page IRQ exit code >>BFD3

**$FFC-$FFF9 UNUSED**

- FFEF: These unused bytes are at $4FEC-$4FF9 when
- FF0F: loaded as part of the "PRODOS" file.

**VECTORS**

- FF0A: NMI Vector
- FF0C: Reset Vector
- FF0E: 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
I(20 data blocks I
I plus one index I--->
I block) I
I             I
I LS$2800 I
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
INAMES OF OPEN FILES I
I--------------I
I $BD00 I
I
I BASIC I
I
I INTERPRETER I
I
I (run location) I
I
I--------------I
I $9A00 I
I
I--------------I
I $4800 I
I BI GLOBAL PAGE I
I--------------I
I $4700 I
I
I BASIC I
I
I INTERPRETER I
I
I (load location) I
I
I--------------I
I $2400 I
I BI RELOCATOR I
I--------------I
I $2000 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.
## BI Relocator -- V1.1.1 -- 18 JUN 84

**Next Object Addr:** 2000

### ADDR DESCRIPTION/CONTENTS

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td><strong>Module Starting Address</strong></td>
</tr>
<tr>
<td></td>
<td>*************************</td>
</tr>
<tr>
<td></td>
<td>* PRODOS BASIC INTERPRETER RELOCATOR</td>
</tr>
<tr>
<td></td>
<td>* LOADED AS THE FIRST TWO BLOCKS</td>
</tr>
<tr>
<td></td>
<td>* OF BASIC.SYSTEM AT $2000.</td>
</tr>
<tr>
<td></td>
<td>* THIS ROUTINE MOVES THE BASIC</td>
</tr>
<tr>
<td></td>
<td>* INTERPRETER TO $9A00-8BCFF.</td>
</tr>
<tr>
<td></td>
<td>* FOR PRODOS VERSION 1.1.1</td>
</tr>
<tr>
<td></td>
<td>* (BASIC Version Number is 1.1.1)</td>
</tr>
<tr>
<td></td>
<td>* Modify date is 18 JUN 84)</td>
</tr>
</tbody>
</table>

### ********** Zero Page Addresses **********

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>&quot;FROM&quot; POINTER FOR COPY</td>
</tr>
<tr>
<td>0001</td>
<td>&quot;TO&quot; POINTER FOR COPY</td>
</tr>
<tr>
<td>0002</td>
<td>&quot;FROM&quot; POINTER FOR COPY</td>
</tr>
<tr>
<td>0003</td>
<td>&quot;TO&quot; POINTER FOR COPY</td>
</tr>
<tr>
<td>0035</td>
<td>CSWL VECTOR</td>
</tr>
<tr>
<td>0036</td>
<td>KSWL VECTOR</td>
</tr>
<tr>
<td>006F</td>
<td>APPLESOFT START OF STRINGS</td>
</tr>
<tr>
<td>0073</td>
<td>APPLESOFT HIMEM</td>
</tr>
<tr>
<td>00F2</td>
<td>APPLESOFT TRACE FLAG</td>
</tr>
</tbody>
</table>

### ********** External Addresses **********

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>0200</td>
<td>PATHNAME BUFFER</td>
</tr>
<tr>
<td>0280</td>
<td>PREFIX BUFFER</td>
</tr>
<tr>
<td>0281</td>
<td>START OF PREFIX NAME</td>
</tr>
<tr>
<td>0300</td>
<td>WARMSTART VECTOR</td>
</tr>
<tr>
<td>0303</td>
<td>COLDSTART VECTOR</td>
</tr>
<tr>
<td>030F</td>
<td>BRK HANDLER ADDRESS</td>
</tr>
<tr>
<td>031F</td>
<td>RESET HANDLER ADDRESS</td>
</tr>
<tr>
<td>0333</td>
<td>POWER-UP BYTE</td>
</tr>
<tr>
<td>03F5</td>
<td>APPLESOFT &amp; VECTOR</td>
</tr>
<tr>
<td>03FB</td>
<td>CTL-Y VECTOR</td>
</tr>
</tbody>
</table>

### ********** Basic Global Page **********

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>BC7A</td>
<td>BASIC INTERPRETER VERSION NUMBER</td>
</tr>
<tr>
<td>BE00</td>
<td>BASIC INTERPRETER ENTRY POINT</td>
</tr>
<tr>
<td>BE03</td>
<td>CI COMMAND SCANNER (SYNTAX)</td>
</tr>
<tr>
<td>BE10</td>
<td>CI HANDLER SHORT VECTOR</td>
</tr>
<tr>
<td>BE2B</td>
<td>KSWL VECTORS FOR EACH SLOT</td>
</tr>
<tr>
<td>BE3C</td>
<td>DEFAULT SLOT NO.</td>
</tr>
<tr>
<td>BE3D</td>
<td>DEFAULT DRIVE NO.</td>
</tr>
<tr>
<td>BEF8</td>
<td>HIMEM</td>
</tr>
</tbody>
</table>

### ********** System Global Page **********

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>BF00</td>
<td>MACHINE LANGUAGE INTERFACE ENTRY</td>
</tr>
<tr>
<td>BF30</td>
<td>LAST DEVICE USED</td>
</tr>
<tr>
<td>BF58</td>
<td>MEMORY MAP</td>
</tr>
<tr>
<td>BF90</td>
<td>MACHINE TYPE FLAGS</td>
</tr>
<tr>
<td>BF99</td>
<td>SLOTS WHICH CONTAINS CARDS WITH ROM</td>
</tr>
<tr>
<td>BF9A</td>
<td>IF Ø, NO PREFIX ACTIVE</td>
</tr>
<tr>
<td>BFFD</td>
<td>INTERPRETER VERSION NUMBER</td>
</tr>
</tbody>
</table>

### ********** ROM Addresses **********

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>EB00</td>
<td>APPLESOFT ENTRY POINT</td>
</tr>
<tr>
<td>FA59</td>
<td>BRK HANDLER</td>
</tr>
<tr>
<td>FB02</td>
<td>INIT SCREEN, MONITOR, ETC.</td>
</tr>
<tr>
<td>FC5B</td>
<td>CLEAR SCREEN, HOME CURSOR</td>
</tr>
<tr>
<td>FE0D</td>
<td>STANDARD CHARACTER OUT</td>
</tr>
<tr>
<td>FE84</td>
<td>SET NORMAL CHARACTER ATTRIBUTE</td>
</tr>
</tbody>
</table>

### 2000 ********** Basic Interrelocator Entry **********

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td>JUMP OVER STARTUP FILENAME &gt;&gt;2047</td>
</tr>
<tr>
<td>2006</td>
<td>STARTUP FILENAME LENGTH (7)</td>
</tr>
<tr>
<td>2007</td>
<td>'STARTUP'</td>
</tr>
<tr>
<td>200E</td>
<td>ALLOW FOR 64 CHAR FILENAME</td>
</tr>
<tr>
<td>2047</td>
<td>$00 -- $2400</td>
</tr>
<tr>
<td>2048</td>
<td>$22 -- $9A00</td>
</tr>
<tr>
<td>2055</td>
<td>COPY 35 PAGES</td>
</tr>
<tr>
<td>2056</td>
<td>COPY INTERP TO HIGH MEMORY AT $9A00 &lt;2BC4&gt;</td>
</tr>
<tr>
<td>205D</td>
<td>PAGE FOLLOWING INTERP IMAGE IS...</td>
</tr>
<tr>
<td>205F</td>
<td>BASIC GLOBAL PAGE IMAGE</td>
</tr>
<tr>
<td>2061</td>
<td>COPY THAT TO 5BE0 &lt;2BC4&gt;</td>
</tr>
<tr>
<td>2064</td>
<td>TO GET 40-COL DISPLAY, SEND A CTRL-U</td>
</tr>
<tr>
<td>2066</td>
<td>OUT THE NORMAL OUTPUT VECTOR. &lt;FEFD&gt;</td>
</tr>
<tr>
<td>2069</td>
<td>SET NORMAL CHARACTER ATTRIBUTE &lt;FE84&gt;</td>
</tr>
<tr>
<td>206C</td>
<td>INITIALIZE SCREEN/WINDOW &lt;FB02&gt;</td>
</tr>
<tr>
<td>206F</td>
<td>CLEAR SCREEN/HOME CURSOR &lt;FC5B&gt;</td>
</tr>
</tbody>
</table>
2127 NO >>214E
212F ERROR? >>218B
2136 BACKSCAN PREFIX FOR "\" (0280)
2138 AND COUNTER THEM IN $218E (213E)
213F FOR A COUNT OF SUBLEVELS >>2136
2146 MORE THAN JUST VOLUME NAME? >>216F
2148 NO, MLI: SET PREFIX <BP00>
214E MLI: ONLINE <BP00>
2154 ERROR? >>218B
2156 GET VOLUME LENGTH (0281)
2158 NULL THERE? >>218B
215F ADD ONE TO NAME LENGTH (028B)
2164 AND PREFIX IT IN A "\" (0281)
2167 MLI: SET PREFIX <BP00>
216D ERROR? >>218B

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

216F MLI: GET FILE INFO <BP00>
2172 FIND "STARTUP" FILE
2175 ERROR? >>218B
217A SAVE LENGTH OF STARTUP FILE NAME (2236)
217D COPY NAME TO $200 (2006)
2186 FIRST COMMAND WILL BE "STARTUP"
218B CHECK NUMBER OF SUBLEVELS (223E)
2190 MORE THAN JUST VOL? >>219B
2192 MLI: SET PREFIX <BP00>
2196 ANY STARTUP FILE NAME? (2236)
219B YES, SKIP MESSAGE >>21Cl
219D SET TRUE KSWSL <2299>
21A2 PRINT "FRODOTs BASrIC 1.1" (2267)
21AD PRINT 'COPYRIGHT ... (22B3)
21BE SKIP THREE LINES

******** FINISH UP AND GO TO BI **********
Beneath Apple ProDOS Supplement

BI Relocator -- V1.1.1 -- 18 JUN 84
ADDR 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 (0206)
2217  FOLLOWED BY A RETURN
2219  RETURN

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

221A  CSWL (2BD4) INTERCEPT ADDR
221C  KSWL (2B09) INTERCEPT ADDR

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

2231  SET PREFIX PARM LIST
2232  FOR PREFIX AT $2234

2234  NULL PREFIX
2235  "/

2236  SAVED LENGTH OF STARTUP FILE NAME

2237  ONLINE PARM LIST
2239  PUT VOLUME NAME AT $281

223B  SET PREFIX PARM LIST
223C  PREFIX IS AT $280

223E  NUMBER OF SUBLEVELS IN PREFIX +1

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

22A3  ******** $22A3-$23FF NOT USED ***************
### PAGE ADDRESSES

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/ CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>9A00</td>
<td>MODULE STARTING ADDR</td>
</tr>
</tbody>
</table>

**LIBRARY**

- **PRODOS BASE**: (CML)
- **THIS**: BLOCK VECTOR (KSWL)
- **IT PE**: FOR R AND LOOP COUNTER
- **HAND**: AND LOOP COUNTER
- **VERSION 1**: ESOFTE VARIABLES
- **DISTRIB**: NUMBER

**ADDRESS OF PROGRAM PTR**

- **ZERO**: (START OF VARS)
- **SCREEN LINE**: (ARRAY VARS PTR)
- **MONITOR PROMPT**: (FREE AREA PTR)
- **CRT DISPLAY VECT**: (END OF STRINGS)
- **SCRATCH POINTER**: (POINT TO APP
- **APPLOSOFT**: LNI
- **HIM**: ADD

**EXTERNAL ADDRESSES**

- **0100**: START OF 6502 STACK
- **0200**: KEYBOARD INPUT LINE BUFFER
- **03F4**: POWER:ON RESET FLAG

**BI GLOBAL PAGE**

- **BE46**: EXTERNAL COMMAND ENTRY TO BI
- **BE5C**: PRODOS ERROR CODE
- **BE10**: OUTPUT VECTORS FOR ALL SLOTS
- **BE30**: CURRENT OUTPUT VECTOR
- **BE32**: CURRENT INPUT VECTOR
- **BE34**: PRODOS INTERCEPT VECTORS (INPUT/OUTPUT)
- **BE38**: BI'S INTERNAL REDIRECTION VECTORS
- **BE3C**: DEFALUT SLOTH
- **BE3D**: DEFALUT DRIVE
- **BE3E**: A REGISTER SAVE AREA
- **BE3F**: X REGISTER SAVE AREA
- **BE40**: Y REGISTER SAVE AREA
- **BE41**: TRACE FLAG (APPLOSOFT TRACK ON/OFF)
- **BE42**: IMMEDIATE COMMANDS=0, DEFERRED=1
- **BE43**: EXEC FILE ACTIVE=S00
- **BE44**: READ FILE ACTIVE=S00
- **BE45**: WRITE FILE ACTIVE=S00
- **BE46**: READING PREFIX ACTIVE=S00
- **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

APPLESOF ROM LOCATIONS

RESET I/O ROMS

MONITOR ROM LOCATIONS

COUT VECTOR

BASIC INTERPRETER LOAD POINT

ENTRY POIN IS AT $8BF1, WARMDO

REMOVE KS/L/CSW/L INTERCEPTS

REPLACE KS/L/KSW/L WITH CURRENT (BE38)

ACTUAL DEVICE DRIVER VECTORS

RESET MODE/SET BI INTERCEPTS

SET IMMEDIATE COMMAND MODE

AND GO SET I/O VECTORS <9F76>

KS/L/H ALREADY SET?

NO? THEN CHECK CSW/L >>9A26

YES, CONTINUE >>9A3

CSW/L/H ALREADY SET?

YES, CONTINUE >>9A3

NO, SAVE CURRENT INTERCEPTS FIRST >>9A8D

OUTPUT INTERCEPT: MODE = 0

(SAME BIT)

KEYBOARD STROBE CLEAR

APPLESOF PRINT DECIMAL NUMBER

APPLESOF SET NORMAL CHAR

MONITOR CLEAR TO EOL

MONITOR READ KEY (NO CURSOR)

COUT VECTOR

BASIC INTERPRETER LOAD POINT

ENTRY POINT IS AT $8BF1, WARMDO

REMOVE KS/L/CSW/L INTERCEPTS

REPLACE KS/L/KSW/L WITH CURRENT (BE38)

ACTUAL DEVICE DRIVER VECTORS

RESET MODE/SET BI INTERCEPTS

SET IMMEDIATE COMMAND MODE

AND GO SET I/O VECTORS <9F76>

KS/L/H ALREADY SET?

NO? THEN CHECK CSW/L >>9A26

YES, CONTINUE >>9A3

CSW/L/H ALREADY SET?

YES, CONTINUE >>9A3

NO, SAVE CURRENT INTERCEPTS FIRST >>9A8D

OUTPUT INTERCEPT: MODE = 0

(SAME BIT)
BASIC Interpreter (B1) -- V1.1.1 -- 18 JUN 84 NEXT OBJECT ADDR: 9A2F

ADD! DESCRIPTION/CONTENTs

9A2F ** "CHAR"**

9A32 NO... >>5L

9A34 ELSE, SAVE...

9A38 CHECK START? (9F61)

9A3B (APPLESOFT5A)

9A44 NOT TRAC'r X REG (BE3F)

9A46 ELSE, SETK FOR $DB12 AS RETURN ADDR ($103)

9A48 GET SET (TRACE, PRINTING $110)

9A50 RESTORE >9509 ? >>9A68

9A51 AND GO TO|DEFERRED MODE=4

9A54 PRINT THE "#" (9F61)

9A57 TAKE REG (BE3F)

9A5A NO, OTHER OUTPUT HANDLER >>B7F1

9A5C NO, >>5L

9A5E NO, ALL >>5L

9A68 HAS HORIZ? >>9A74

9A6B YES... >>5L

9A6D ELSE, >>9A74

9A6F MUST BE NTAL CURSOR POSN CHANGED?

9A72 RESTORE >>9A69

9A75 PATHNAME XING IN PATHNAME BUFFER? (SCBD)

9A77 ELSE, WE ALPHAD...

9A7A A BB98 THERE... >>9A74

9A7B ARE RECURSING INFINITELY, EXIT!

9A7C TRACING AFTER ALL,Restore X (BE3F)

9A7E THEN FALL THRU TO EXIT (9F61)

9A82 PUT BACK

9A87 OUTPUT TAOCHC_OUTPUT CHAR AND EXIT ***************

9A7A WAS IT A...

9A87 NO, EXIT REAL CSWL/KSWL VECTORS <<9A09

9A8E ELSE, WHERE CHARACTER <FDE0>

9A82 YES >>9A50 RETURN?

9A84 NO, CLEAR NOW >>9A5D

9A87 FORCE AP8AppleSOFT TRACING?

9A88 RESTORE >>3

9A8D ---------- MY TRACE FLAG (PSUEDO_TRACE NOW) (BE41)

9A8B ---------- AP8 TO TRAC'r FOR MY BENEFIT ONLY

9A8D --------

9A8E COPY KSW SAVE ACTUAL IN/OUT VECTORS ***************

9A98 AND CSWL

9A9A IN BI GL

<<a TO VECIN

<<b TO VECOUT

<<cAL PAGE (BE31)

9A9D **

9A9E ERROR HANDLER ***********************

9A9E ** ERROR =3, "NO DEVICE CONNECTED"

9A9F MAIN ENTRY: STORE ERROR CODE <<9F0

9A9F AND IN APPLESOFT ONERR

9A9E CHECK BI STATE (BE42)

9A9F MEMORIZETH AT IT'S IMMEDIATE MODE

9A9D SET A HIGH FILE LEVEL FOR NON-EXEC FILES (BE94)

9A9D NO ACTIVE READ/WRITE FILES OR PREFIX READ (BE44)

9A9E CLOSE ALL OPEN FILES AT OR ABOVE (BE8E)

9A9F FILE LEVEL = <<9F

9A9F MLI: CLOSE <ALL> <<9E70

9A9E ERROR? >>9B27

9A9B WRITE ANY DATA HAVE BUFFERED <<9A00

9A9B ERROR? >>9B27
BASIC Interpreter (BI) -- V1.1.1 -- 18 JAN 84  NEXT OBJECT ADDR: 9B9A
---------------------------------------------------------------------
9B1A  PUT FILE LEVEL BACK TO ZERO
9B22  NOW FLUSH ALL OPEN FILES
9B24  MLI: 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 CSWL/KSWL INTERCEPTS <9AA3>
9B37  GET ERROR CODE (BE0F)
9B3B  BASIC ONERR ACTIVE? THEN GO HANDLE IT >>9B44
9B3E  NO, JUST PRINT ERROR MESSAGE <BEIC>
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 **********
9B5A  CLEAR APPLESOFT ERRNUM
9B5C  WILL LOOK FOR "I" FROM APPLESOFT
9B61  SET NORMAL VIDEO IN APPLESOFT <F273>
9B64  RESTORE TRUE CSWL/KSWL <9A00>
9B67  TRY TO WRITE BUFFERED DATA <FF94>
9B6A  RESET MODE/SET UP BI'S INTERCEPTS <9A17>
9B6D  RESTORE INTERCEPTS <9F6C>
9B7B  GO TO PROCESS IMMED. INPUT REQUEST >>9ABA
9B73  ********** INPUT INTERCEPT: MODE=4 OR 8 **********
9B76  SAVE REGISTERS <9F62>
9B7A  PREFIX INPUT ACTIVE? (BE46)
9B79  NO >>9B7E
9B7A  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 >>9BAF
9B8B  YES, GET PROMPT CHARACTER
9B8D  IT BETTER NOT BE A "I"
9B8F  IT IS, RETURN TO IMMEDIATE MODE >>9B58
9B91  ELSE, SET TRUE CSWL/KSWL <9A00>
9B94  AND PASS CALLER'S AREG TO REMOVE CURSOR (BE3E)
---------------------------------------------------------------------
BASIC Interpreter (BI) -- V1.1.1 -- 18 JAN 84  NEXT OBJECT ADDR: 9B97
---------------------------------------------------------------------
9B97  RESTORE Y-REGISTER (BE40)
9B9A  REMOVE CURSOR AND GET A KEYPRESS <FD1D>
9B9D  BACKSPACE?
9B9F  NO, EXIT BI >>9BAC
9BA1  YES, CHECK PROMPT
9BA3  IF ITS A >>...
9BAA  THEN EXIT WITH THE BACKSPACE >>9BAA
9BAC  ELSE, IF AT START OF LINE, REPROMPT >>9B94
9BAA  MIDDLE OF LINE, RETURN A BACKSPACE
9BAC  EXIT BI TO CALLER >>9BA6
9B9F  ************ READ EXEC FILE **********************
9BAF  REMOVE CURSOR FROM SCREEN
9B11  CHECK PROMPT CHARACTER
9B1B  IF ITS A >>...
9B15  DO THINGS DIFFERENTLY >>9BF2
9B17  CHECK KEYBOARD (C000)
9B1A  NO KEY READY? >>9BCD
9B1C  GOT A KEY, IS IT CONTROL-C?
9B1E  NO, IGNORE IT >>9BCD
9B1C  YES, CLOSE EXEC FILE <B2FB>
9B1D  IMMEDIATE MODE? (BE42)
9B1E  NO >>9C01
9B1C  YES, CLEAR KEYBOARD STORE (C018)
9B1B  AND GO START NEW LINE >>9C91
9B1D  SET UP FOR EXEC LINE READ <9DA8>
9B1D  READ A LINE TO $200 <9C5C>
9B1E  ERROR? >>9BFA
9B1F  SAVE REGISTERS <9F62>
9B28  HOP INTO LOOP >>9BDE
9B2A  ---
9B2B  BACKSCANNING $200 BUFFER (D200)
9B2D  FORCING THE MSG ON
9B2E  RESTORE TRUE CSWL/KSWL <9A00>
9B2F  GO PROCESS COMMAND LINE <9A5D>
9B3C  CHECK COMMAND NUMBER (BE53)
9B3F  IMMEDIATE EXIT? IF NOT, GET NEXT LINE >>9BCD
9BF1  RETURN

********** HANDLE EXEC PROMPT > **********
9BF2  GET SET TO READ EXEC LINE <9DA8>
9BF5  READ SINGLE CHARACTER PER CALL <9C40>
9BF8  NO ERRORS, EXIT TO CALLER NOW >>9BF1

********** EXEC ERROR RECOVERY ******************
Beneath Apple Pro

DOS Supplement

BASIC Interpreter

ADDR DESCR

9BFA CLOSE "B1:" V1.1.1 -- 18 JUN 84 NEXT OBJECT ADDR: 9BFA

9BFD NO REAL ERRORS
9BFF NO, RE

9CB0 ELSE, EXEC FILE <B245>
9CB3 GET CUTOFF "END OF DATA"?
9CB5 IF IN AL ERROR THEN >>9C13
9CB7 ELSE, OR JUST STOP EXECUTING
9CB8 AND RETURN HORIZONTAL POSITION
9CBF RETURN ID LINE, PASS SCREEN CHAR BACK >>9C0E
9C0E GET SCCHANGED PROMPT TO "J"
9C10 AND EXIT WITH A BACKSPACE
9C13 REAL E

SEEN CHARACTER UNDER CURSOR

9C16 **********THRU KSwL TO GET REAL KEYPRESS >>0038
9C18 IF ITS** INPUT FILE ACTIVE ***********************
9C1C THEN F
9C1F ELSE, IFMT
9C24 CHECK A "["...
9C27 NO KEYSET TO IMMEDIATE MODE >>9B58
9C29 GET A REMOVE CURSOR FROM SCREEN (BE3E)
9C2B NO, NO KEYWORD (C008)
9C2D CLEAR SCREEN >>9C31
9C30 RETURN KEY, IS IT CONTROL-C7
9C32 MORE IT >>9C31
9C36 GET PROTOBE AND EXIT TO CALLER (C018)
9C38 IS THE
9C3B YES >>
9C3E NO, IS THAT AGAIN
9C47 YES, IS A DIRECTORY FILE? (BE47)
9C49 ELSE, BC95
9C4F ERROR PROMPT = "?"
9C4A RETURN READ A SINGLE BYTE AT A TIME >>9C4C
9C4C READ ENTIRE LINE <9C67>
9C4D READ ID >>9C13
9C4E ERROR?
9C4F RETURN

9C48 **********SAVE <<9C13
9C4B IN L "**READ NEXT BYTE OF FILE ***********************
9C50 SET U
9C53 MLII: CURRENT READ/WRITE COUNT (BED9)
9C55 ERROR: CWD VALUE (B5F)
9C5A PUT C OF TO READ ONE BYTE (BED9)
9C5B GET FREAD <B378>
9C5D AND RE >>9C66
9C65 RETURN JUST BACK TO MAXIMUM AGAIN (B5F)
9C66 RETURN'S CHARACTER ON $200 LINE (BED7)
9C70 RETURN THAT TO CALLER (0208)

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

ADDR DESCRIPTION/CONTENT

9C67 ********** READ NEXT LINE OF FILE ***************
9C67 REMOVE CURSOR FROM SCREEN (BE3E)
9C68 ----
9C6E MLII: READ <BE70>
9C71 ERROR? >>9C66
9C73 IF LENGTH ACTUALLY TRANSMITTED (BEDB)
9C76 NOTHING? >>9C6E
9C79 CHAR L ATTENTION STRING TO DATA (BED7)
9C7D FETCH LAST BYTE OF LINE (01FF)
9C82 IS IT A RETURN CHARACTER?
9C84 NO, LEAVE LINE ALONE >>9C66
9C86 YES, WAS U KEYWORD GIVEN? (BE57)
9C88 YES, LEAVE IT BE >>9C6E
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 >>9C6E
9C99 ELSE, REMOVE CURSOR FROM SCREEN (BE3E)
9CA SET 80 COLUMNS
9CAB MLII: GET MARK <BE70>
9C9B ERROR? >>9D1F
9C9C ARE WE AT BEGINNING OF THIS FILE? (BEC8)
9C9D NO, CONTINUE >>9CDF
9C9E YES, CAT FLAG = 2
9C9F READ DIRECTORY HEADER <B15D>
9C9A ERROR? >>9D1F
9CBB REF NUM TIMES 32 (BED6)
9CC SET THE L VALUE OF THIS DIR FILE IN (BCF)
9CCA THE OPEN FILE LIST TO THE ENTRY LENGTH (BCB8)
9CCD AND THE NUMBER OF ENTRIES PER BLOCK (B008)

********** DIRECTORY NAME **********
9CC0 GO FORMAT NAME OF DIRECTORY <B3B8>
9CC3 STORE THE LENGTH OF LINE AT $200
9CC5 PUT A RETURN CHAR AT END OF LINE
9CC6 PUT A RETURN TO CALLER
9CCD RETURN

9CDE GET CAT FLAG (BE4F)
9CE2 IF ZERO, GO PROCESS INDIVIDUAL ENTRIES >>9D22
9CE4 IF MINUS, D0 SAME COUNT LINE OR EXIT >>9C9F
9CE6 POSITIVE, ASSUME NULL LINE WANTED
9CE8 DROP CAT FLAG BY ONE (BE4F)
BASIC Interpreter (BI) -- V1.1.1 -- 18 JUN 84
NEXT OBJECT ADDR: 9CE8

ADDDR DESCRIPTION/CONTENTS

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... <9FB0>

9CF5 LINE LENGTH IS 80

9CF7 GO RETURN IT TO CALLER >>9CD3

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

9CF9 DO SUMMARY LINE?

9CFA NO, JUST EXIT (ALL DONE) >>9D1C

9CFD YES, DROP CAT FLAG SO EXIT NEXT TIME (BE4F)

9DD2 CLEAR READ/WRITE COUNT (BED9)

9DDA MLI: READ <BE70>

9DDD 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 >>9C75

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)

9D28 *32

9D2F USE AS INDEX TO GET ENTRY LENGTH (BCF)

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 (BC7)

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 $281 <A4C4>

9D64 AND RETURN IT TO CALLER >>9CF5

--- End of BASIC Interpreter Code ---
BASIC Interpreter (BI) — VI.1.1 — 18 JUN 84  NEXT OBJECT ADDR: 9DE6

ADDR  DESCRIPTION/CONTENTS

9DE6  RETURN

9DE7  ---

9DE8  NULL LINE? >>9DF6
9DE9  NO, PUT BACK TRUE CSWL/KSWL <9A00>
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)
9E01  CHECK PROMPT
9E03  CHECK TO SEE IF WE ARE IN "IF", >>9E11
9E06  "PRINT", "LIST", OR "CALL" STATEMENTS >>9E11
9E09  OF AN APPLESOFT PROGRAM >>9E11
9E0B  IF NOT, EXIT TO CALLER... (BE44)
9E0E  WITH CHARACTER ECHOED TO SCREEN >>9A74
9E11  GET INDEX TO TEMPORARILY BUFFERED CHAR (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 <9EE6>
9E26  ERROR? >>9DE8
9E28  RESTORE REGISTERS <9F6C>
9E2B  AND EXIT ANYWAY

9E2C  *********** OUTPUT INTERCEPT: MODE = 4 **********************

9E2C  (INITIAL ENTRY FOR A RUNNING PROGRAM)
9E2D  (FLUSH OUT NON COMMAND LINES)

9E2F  PRINTING A "#"? (9F61)
9E30  NO >>9E49
9E31  YES, SAVE X REGISTER (BE3F)
9E35  RETURN ADDR IS IN APPLESOFT... (0103)
9E3B  TRACE ROUTINE...
9E3C  AT $D812? (0104)
9E40  YES >>9E66
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 >>9E66
9E56  NO, PRINTING A RETURN CHAR?
Beneath Apple ProDOS Supplement

BASIC Interpreter (B1) -- 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? >>9EFL
9EC6 OR END OF LINE? >>9EE6
9ECE NEITHER, DECREMENT STRING SPACE CTR (BE49)
9ED1 OK >>9EE6
9ED3 COMPUTE SIZE OF FREESPACE IN PAGES
9ED7 AT LEAST 3 FILES AVAILABLE?
9ED9 YES >>9EE5
9EDB NO, WRITE BUFFERED DATA <9FF4>
9EE6 AND THEN GARBAGE COLLECT <A044>
9EE3 COMPUTE FREE SPACE NOW
9EE5 AND SAVE IN STRING SPACE CTR (BE49)
9EEA GET NEXT TOKEN
9EEC --
9EED JUMP BACK INTO APPELSOFT TO EXECUTE IT >>D820
9EFL STORE TOKEN IN PROMPT
9EF4 LOOK UP TOKEN IN BI'S TOKEN TABLE (B7F9)
9EF7 ITS NOT ONE BI IS INTERESTED IN >>9EE6
9EF9 IT IS INTERESTING, CHANGE BRANCH <<9EF5
9EFC AND JUMP TO ONE OF THE FOLLOWING: >>9EFL

9EF6 IF OR PRINT: PROMPT = 0
9F00 CLEAR OUT LAST CHAR SAVEAREA (BE4C)
9F03 GO TO MODE = C NEXT TIME THRU (B8E3)
9F06 (BEGIN LOOKING FOR COMMANDS) (BE3B)
9F0F NOW GO PROCESS THE IF OR PRINT >>9F2E

9F11 LIST: PROMPT = 1
9F13 (DON'T LOOK FOR COMMANDS NOW)
9F15 GO DO IT >>9F2E

9F17 CALL: PROMPT = 2
9F19 (DON'T LOOK FOR COMMANDS NOW)
9F1B GO DO IT >>9F2E

9F1D LET: DECREMENT STRING CTR
9F1E AND GO BACK FOR NEXT TOKEN >>9EC6

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

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

82
BFB0  ********** UNPACK ERROR MESSAGE ***********************

BFB0  NOTHING IN BUFFER AT FIRST
BFB6  GET A NIBBLE FROM PACKED MSG  <BFD2>
BF9  NON-ZERO, COMMON CHARACTER  >>BFC3
BF8  IF ZER0, GET NEXT NIBBLE  <BFD2>
BFD  AND CONVERT TO UNCOMMON CHAR INDEX
BF9  ---
BFC1  GET THE LETTER THIS NIBBLE REPRESENTS  (BA4B)
BFC4  ZERO? THEN END OF MESSAGE  >>BFD1
BFCA  GET INDEX INTO OUTPUT BUFFER  (BE4B)
BF9  AND STORE THE CHARACTER THERE  (0201)
BFF  BUMP INDEX  (BE4B)
BF0  AND CONTINUE  >>BFB6
BFD  RETURN

BFD2  ********** UNPACK MESSAGE BYTE ***********************

BFD2  GET NEXT MSG BYTE  (BA4B)
BDF  WORKING ON SECOND NIBBLE?  >>BFD9
BDF  NO, TAB INDICATOR?  >>BFD0
BDF  NO, ISOLATE HIGH NIBBLE
BFD  NEXT TIME GET LOW NIBBLE
BDF  RETURN

BDF  ---
BDF  GET TAB POSITION  (BA4B)
BFE  AND BUMP OUTPUT PTR ACCORDINGLY  (BE4B)
BFE  THEN GO BACK FOR NEXT NIBBLE  >>BFD2

BFE9  BUMP BYTE PRT FOR NEXT TIME
BFEA  ISOLATE LOW NIBBLE
BFE  NEXT TIME GET HIGH NIBBLE
BFE  RETURN

BFEE  ********** WRITE ONE BUFFERED BYTE *****************

BEFF  SET UP COUNT OF 0001
BEF2  AND JUMP INTO ROUTINE BELOW  >>A007

BFF4  ********** WRITE BUFFERED DATA/TEST ERROR *********

BFF4  WRITE BUFFERED DATA  <A000>
BFF7  OK? THEN EXIT  >>A81C
BFFA  ERROR, POP OUT OF THIS SUBROUTINE
BFFD  AND GO TO ERROR HANDLER  >>9AF0

BFD  ********** WRITE ALL BUFFERED DATA ***********************

BFD  ---
BDD  GET BUFFERED DATA COUNT  (BE4A)
BDF  NONE BUFFERED?  >>A81B
BDF  STORE BUFFERED DATA COUNT IN RW PARMS  (BE9)
BDF  MLI: WRITE  <BE70>
BDF  NOTHING BUFFERED NOW, COUNT=0  (BE4A)
BDF  ERROR?  >>A81C
A81  NO, EXIT
A81C  RETURN

A81D  ********** SPECIAL GARBAGE COLLECT ***********************

A81D  (PULL OUT STRING CONSTANTS ALSO)
A81D  DO GARBAGE COLLECTION NORMALLY FIRST  <A844>
A820  ERROR?  >>A843
A824  START OF STRING AREA = PROGRAM START PRT  (BC84)
A82C  USE GENERAL PURPOSE BUFFER  (ABOVE HIMEM)
A82E  FOR A GARBAGE COLLECT WORKAREA  (BC7D)
A833  IT IS 3+1 PAGES IN LENGTH  (BC7E)
A83B  END OF STRING AREA IS AT END OF FREEAREA  (BC86)
A840  GO COLLECT CONSTANT STRINGS NOW  <A805>
A843  THEN EXIT

A844  ********** "FRE" COMMAND ***********************

A844  (FAST APPLESOFT STRING GARBAGE COLLECTION)

HIMEM  ---

GENERAL PURPOSE BUFFER  ---
(TOP OF OLD STRINGS)  ---

NEW STRINGS BUILDING  ---
(DOWN)  ---

\///\///\///\///\///\///\///\///\///\///

OLD STRINGS  ---

FREE AREA  ---

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.
<table>
<thead>
<tr>
<th>BASIC Interpreter (BI) -- V1.1.1 -- 18 JUN 84 NEXT OBJECT ADDR: A0F6</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADDR</td>
</tr>
<tr>
<td>A0F7</td>
</tr>
<tr>
<td>A0F7</td>
</tr>
<tr>
<td>A0F8</td>
</tr>
<tr>
<td>A102</td>
</tr>
<tr>
<td>A108</td>
</tr>
<tr>
<td>A10A</td>
</tr>
<tr>
<td>A111</td>
</tr>
<tr>
<td>A113</td>
</tr>
<tr>
<td>A117</td>
</tr>
<tr>
<td>A11B</td>
</tr>
<tr>
<td>A11E</td>
</tr>
<tr>
<td>A123</td>
</tr>
<tr>
<td>A125</td>
</tr>
<tr>
<td>A129</td>
</tr>
<tr>
<td>A12A</td>
</tr>
<tr>
<td>A12B</td>
</tr>
<tr>
<td>A12C</td>
</tr>
<tr>
<td>A12D</td>
</tr>
<tr>
<td>A12D</td>
</tr>
<tr>
<td>A12D</td>
</tr>
<tr>
<td>A130</td>
</tr>
<tr>
<td>A132</td>
</tr>
<tr>
<td>A136</td>
</tr>
<tr>
<td>A139</td>
</tr>
<tr>
<td>A13E</td>
</tr>
<tr>
<td>A140</td>
</tr>
<tr>
<td>A143</td>
</tr>
<tr>
<td>A145</td>
</tr>
<tr>
<td>A146</td>
</tr>
<tr>
<td>A147</td>
</tr>
<tr>
<td>A151</td>
</tr>
<tr>
<td>A154</td>
</tr>
<tr>
<td>A158</td>
</tr>
<tr>
<td>A15A</td>
</tr>
<tr>
<td>A15C</td>
</tr>
<tr>
<td>A15D</td>
</tr>
<tr>
<td>A164</td>
</tr>
<tr>
<td>A166</td>
</tr>
<tr>
<td>A16A</td>
</tr>
</tbody>
</table>
Beneath Apple ProDOS Support

A16C POINT TO ARRAY (BC7C)
A176 MSB TO X REG STRING OUT C4
A17D CHECK TYPE OF WHAT UNDER HIMEM WORKAREA (BC7C)
A182 SKIP INTEGER AN intercourse.
A186 GET NUMBER OF E C4 "THIS STRING"
A188 *2 TO SKIP SIZED SAVED AREA? (IE PTR
A189 +5 TO SKIP FIXED THERE THEN >A ALSO
A19D POINT TO FIRST C4 "SIZED STRING"
A191 READY TO ROLL, PING
A194 RETURN ART PTR BY LEN (BC7B IN VARIABL
A195 ********** COPY STRING START PTR
TO MAKE ROOM P PTR
TO HIMEM, COL, LENGTH STRING?
STRING AREA do >A1EE ON

A1A5 COPY N+1 PAGES >E7 (BC82)
A182 EXIT WHEN FINISH
A188 ********** PULL TACK STRING
STRING START
A188 IS STRING BELOW?
A18B YES, ITS STILL
A18D ELSE, POINT TO:
A1A4 $3A/$3B -> ST
A1C8 DROP STRING ST
A1D4 UPDATE STRING
A1D8 FIX UP MSB OF:
A1DD AND OF VARIABL:
A1E1 IS THIS A NULL?
A1E3 YES, NO MOVE T
A1E6 ---
A1E7 ELSE, COPY STR
A1EE ---
A1EF OUT OF FREE SPA
A1F4 RETURN TO CALL

A1F5 ********** ALLOC

A1F5 NEED 4 PAGES

********** GENERAL PURPOSE ALLOCATE **********
BASIC Interpreter (BI) -- v1.1.1 -- 18 JUN 84 NEXT OBJECT ADDR: A2B4

A2B5 **************** GETBUF: R
   THIS ROUTINE IS A GETA BUFFER **********************
   ENTRY POINT IN CHALLED THROUGH AN EXTERNAL
   CATES A FIXED LOCATION GLOBAL PAGE. IT ALLO-
   BI AND ITS BUFFERING BUFFER BETWEEN THE
   CMPS.
A2B5 ALLOCATE A BUFFER
A2B6 ERROR? >>A30F 1F IF ANY SIZE (A=PAGES) <A1F>
A2BD FIND FIRST PAGE OF
A2C4 GET FILE OPEN COUNTER BUFFER (BB4A)
A2C7 NONE OPEN? >>A231 1 (BE4D)
A2C9 BUMP BUFFER PAGE 81
A2CD TO POINT TO PREV BUFFER BY $400 (BB49)
A2CF BUFFER. (BB49) LSSY ALLOCATED
A2D2 FIND OPEN FILE W/.
A2D7 G0T IT, (BB9) T0 THIS BUFFER (BC93)
A2DA SET FILE BUFFER READ
A2DB THEN SET IT TO NEW LOW IN MEMORY <A352>
A2DB BELOW ALL OTHERS (BUFFER LOCATION <A29B)
A2D7 DO THIS FOR EACH BSEC9)
A2DB THEREBY INSERTING OPEN FILE...
A2ED IS EXEC FILE ACCMPT BLANK BUFFER >>A2D2
A2F0 NO, DONE >>A2FF 12 (BE43)
A2F2 YES,
A2F4 MOVE EXEC BUFFER ON
A2FD AND BUMP UP ABOVS OWN ALSO <A352>
AJF F EXIT TO CALLER T
A300 RETURN

A301 ********** FREEBUF: R
   THIS ROUTINE IS AFREE BUFFER **********************
   ENTRY POINT IN CHALLED THROUGH AN EXTERNAL
   CATES A FIXED LOCATION GLOBAL PAGE. IT FREES
   CATED BY GETBURF BUFFER PREVIOUSLY ALLO-
A301 GET COUNT OF COPY E
A305 INDEX THIS BY 4 PAGES (BE4D)
A306 ADD TO HIMEM MSB D PB PER FILE
A308 SAVE THIS AS TOP CF
A30D THEN SET UP BOTTOM BUFFERS (BB49)
A310 GET OLD ORIGINAL HHM HIMEM MSB (BB4A)
A311 SAME AS THIS ONE? HIMEM (BEFORE ANY BUFFERS) (BEF8)
A315 THEN NOTHING ELSE?
A317 ASSUME NO BUFFERS; GO DO >>A358
A319 ANY EXEC FILE OPEN? REPLACING OLD HIMEM
A31C NO, CONTINUE >>A335 (BE43)
A31E YES, MOVE EXEC BINS
A321 AND MOVE HIMEM TO OLD HIMEM <A2F2>
A323 ELSE, START WITH CMTHN BY $400 >>A341
A326 ANY OPEN FILES? (BBP BUFFER (BB49) 14D)

A329 IF NOT, WE ARE DONE >>A34D
A32B SEARCH FOR OPEN FILE WITH THIS BUFFER (BC93)
A32E NOT IT? >>A34A
A330 G0T 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
A38B PAGE COUNT GONE TO ZERO? (BC93)
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
Beneath Apple ProDOS Supplement

BASIC Interpreter (B1) -- V1.1.1 -- 18 JUN 84 NEXT OBJECT ADDR: A39E

BASIC Interpreter (B1) --

ADDRESS DESCRIPTION/CONTENTS

A39F ************ ADJUST ALL STRING ADDR ***********************

(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
A3AF JUMP INTO THE LOOP >>A3A9
A3A9 ___
A3AB SKIP ONE SIMPLE VARIABLE
A3A0 ----
A3B1 OVERFLOW? >>A3B5
A3B3 YES, BUMP MSB
A3B5 FINISHED WITH SIMPLE VARS?
A3B9 (CHECK BOTH MSB AND LSB OF PTR)
A3BB ----
A3BF YES... >>A3D2
A3BE NO,
A3C0 LOOK AT A SIMPLE VARIABLE
A3C5 SKIP INTEGER AND REAL VARS >>A3AA
A3C7 (DOUBLE CHECK MSB)
A3CB ITS A STRING, POINT TO ITS LEN/ADDR
A3CC ADJUST IT IF NECESSARY <A3F9>
A3CF THEN SKIP OVER IT >>A3AA

A3D2 COPY ARRAYS STARTING LSB
A3D4 (MSB IS IN X REGISTER NOW) (BC81)
A3D7 ----
A3DB FIND A STRING ARRAY <A15C>
A3D8 NO MORE? THEN DONE... >>A40C
A3DD ----
A3E0 ADJUST ITS ADDRESS IF NEED BE <A3F9>
A3E6 SKIP TO NEXT STRING ELEMENT OF ARRAY
A3EE AT END OF THIS ARRAY YET? (BC81)
A3F1 NO... >>A3DD
A3F3 (CHECK MSB ALSO)
A3F7 YES... GO GET NEXT ARRAY >>A3D7

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

A3F9 GET STRING LENGTH
A3F9 IGNOR 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

A40D ************ COMPRESS***************

SIMPLE AND ARRAY VARS

A40D THIS ROUTINE SQUARES UP AGAINST THE HIMEM --

A40D FROM STRING START

A40D PUT THE VARS UNDER (BC92)

A40D ON AN EVEN PAGE BOUND

A40D TO PUT VARS

A40D ROUNDED TO EVEN

A40D ALIGNMENT

A40D STRINGS <A3F7>

A40D PTR (BC98)

A40D 3A BY ONE

A40D FROM HIMEM TO COMPUTE (BC99)

A40D COMBINED VARS/STRINGS

A40D BC9D

A40D IN CASE THEY ARE MOVED

A40D GARBAGE COLLECT P:

A410 ERROR? >>A417

A412 COMPLEATE LENGTH OF VARS SIMPLE AND ARRAY VARS

A417 AND SAVE IT (BC9E)

A427 NEXT, COMPUTE LEN

A42B AND SAVE IT (BC8)

A435 SUBTRACT VARS LEN

A437 TO FIND A PLACE TO

A43A THE STRINGS (START)

A440 $3C/$3D --> PLACE

A447 $3A/$3B --> START

A449 PAGE FREE SPACE

A44F COPY VARS UP AGAIN

A454 STORE START OF VARS

A457 DUMPING PAGE NUMBER

A463 SUBTRACT THIS PTR

A466 TOTAL LENGTH OF ALL

A46A AND SAVE THIS TOO

A46B ALSO, SAVE HIMEM

A471 DONE, EXIT

A472 ************ REEXPAND************

THIS ROUTINE MIGHT

BACK DOWN TO LOW

HIMEM --
>>A53B
A VALUE GIVEN AS SUBTYPE
RT R VALUE TO DECIMAL <<62F>
OVER BIN CODE >>A536
FILE, USE AD VALUE AS SUBTYPE
AT IT TO TWO HEX DIGITS <<612>
Y "=" SIGN
45B OF END OF FILE MARK (0270)
AT LOW TWO BYTES OF EOF <<62F>
ATION DATE/TIME <<570>
RT BLOCKS USED <<62F>
FOR WRITE ACCESS
ED? >>A56C
AD A ""
THRU TO DO LAST MODIFIED DATE/TIME
THEN EXIT TO CALLER

*** FORMAT A DATE/TIME ***********************************************
OFFSET FROM $259 TO FIELD
$201 OFFSET TO DATE/TIME VALUE
CE YEAR ($25A)
ORE IT (BCB5)
CE DAY
ORE IT (BCB4)
CE MONTH
1 = 0 IS NO GOOD >>A5A3
1 > 12 IS ALSO BAD >>A5A3
MONTH (BCB3)
PLY MONTH INDEX BY 3 (BCB3)
VE IT INSTEAD (BCB3)
0 IS NO GOOD >>A5A3
 MUST BE < 99) >>A5B5
WSE, BAD DATE1
IP 6 CHARACTERS ON LINE
INT "NO DATE" (B9E5)
EXIT RIGHT AWAY
OK, GET HOUR ($25C)
UTES ($25B)
8 > 607
A5C3
USE ZERO MINUTES
ET MINUTES (LEFT ZERO FILL) <<60A>
PRINT A ":" ($201)
OUR AGAIN
OR THAN 24 HOURS?
A5D4
ZE ZERO
BASIC Interpreter (BI) -- V1.1.1 -- 18 JUN 84  NEXT OBJECT ADDR: A5D4

ADD DESCRIPTION/CONTENTS

A5D4  10 OR MORE HOURS (TWO DIGITS?)
A5D7  IN ANY CASE, CONVERT HOURS <A62F>
A5DB  IF TWO DIGITS... >>A5DE
A5DD  IF ONE, ADJUST LINE PTR
A5DE  ---
A5E2  CONVERT YEAR (LEFT ZERO FILL) <A60A>
A5E6  GET MONTH INDEX (3) (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 (BCB3)
A632  STORE NUMBER IN ACCUMULATOR (BCAF)
A635  DIVIDE BY 10 <A64D>
A638  GET DIGIT AND CONVERT IT (BCB2)
A63D  STORE IN LINE (0201)
A640  AND DROP LINE INDEX BY ONE
A641  IS QUOTIENT NOW ZERO? (BCAF)
A64A  NO, CONTINUE UNTIL IT IS >>A635

A64C  ELSE, EXIT
A64D  ******* DIVIDE ACCUMULATOR BY 10 *******
A64F  24 BIT SHIFT (3 BYTES)
A651  CLEAR SUM (BCB2)
A654  GO ROL ACCUMULATOR LEFT ONE BIT <AAD7>
A657  ALSO ROL 4TH BYTE OF ACCUM (BCB2)
A65B  IF MSB > 10... (BCB2)
A67665  THEN ADD ONE TO ACCUMULATIVE SUM (BCAF)
A66B  RETURN
A66C  ---
A676  RETURN
A67

A7  ********** SYNTAX; PARSE COMMAND LINE **********
    (ALSO EXTERNAL ENTRY FOR COMMAND STRINGS)
A7

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 <AAB1>
A689  GETFIRST 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 <AEE1>
A699A  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)
A6B7  NO SECONDARY PATH NAME EITHER (0200)
A6BD  CURRENT SLOT = DEFAULT SLOT (BE61)
A663  CURRENT DRIVE = DEFAULT DRIVE (BE62)
A6CB  BUFFER ALLOCATION = HIMEM (BC88)
A6CC  GET LENGTH OF COMMAND NAME (BE52)
A6D8  ALLOW 2 MORE CHARACTERS FOR NOW (BCAA)
A6D3  ARE ANY PARAMETERS PERMITTED? (BE54)
A6D6  NO... MUST BE MON OR NONOM >>A736
A6D8  YES, IN# OR PA#7
A6D9  YES... >>A739
A6DB  ELSE, REPARSE THE COMMAND <AAB1>
A6E0  FOR THIS COMMAND... (BE54)
Beneath Apple ProDOS Supplement

BASIC Interpreter (BI) -- V1.1.1 -- 18 JUN 84 NEXT OBJECT ADDR: A816

ADDRESS DESCRIPTION/CONTENTS

A810 EXTERNAL COMMAND? IF SO GO NOW >>A836
A812 MY OWN COMMAND, "PREFIX"?
A814 YES, GO NOW >>A836
A819 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 <B7DB>
A826 NO ERRORS I HOPE >>A836
A828 ERROR WAS PATH NOT FOUND?
A82A NO, REAL ERROR - SAY SO >>A838
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 ***********************

A83A LOAD B1 CODE FOR "SYNTAX ERROR"
A83B AND RETURN WITH ERROR CONDITION
A83C RETURN

A83D *************** ADD PREFIX TO PATHNAMES ***************

A83D GET SLOT NUMBER (B6E1)
A844 PUT SLOT IN HIGH 3 BITS
A846 DRIVE TO TOP BIT AND SHIFT SLOT DOWN (BE62)
A84E ...TO FORM THE UNIT NUMBER (BEC7)
A853 READ THE PATHNAME PREFIX TO $201 (BEC8)
A85D ML1: ONLINE <BE70>
A868 ERROR? >>A838
A865 DEFAULT DRIVE = PARSED DRIVE (B6E3)
A86B DEFAULT SLOT = PARSED SLOT (B6C3)
A871 PATHNAME1 STARTS WITH "/'/"
A873 THEN ITS ALREADY GOT A PREFIX >>A86E
A876 ELSE, GET LENGTH OF PATHNAME
A87A BUMP IT BY 2 (TO ALLOW FOR '/S')
A882 WITH PREFIX WILL IT EXCEED 64 CHAR?
A88E YES, "SYNTAX ERROR" >>A867
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 (B653)
A8AD "OPEN"?
A8AF YES, DONE NOW! >>A866
A8B1 "APPEND"?
A8B3 YES, DONE NOW! >>A866

BASIC Interpreter (BI) -- V1.1.1 -- 18 JUN 84 NEXT OBJECT ADDR: A885

ADDRESS DESCRIPTION/CONTENTS

A885 "EX"
A887 YES?
A88B ELSE?
A88E GOTO NOW >>A866
A88F COM MAINTAIN PREFIX (B626)
A8BF "GET/PUT" (B63B)
A8C6 IF "IS THIS WITH PREFIX LENGTH (B621)
A8C8 UP TO 64 CHAR?
A8CF ELSE "SYNTAX ERROR" >>A877
A8CE COSE LENGTH (B620)
A8D0 PUT:" PATHNAME2 FORWARD TO MAKE ROOM (B621)
A8D6 "$" IN FIRST
A8D7 DOM "" IN PLACE AND ANOTHER SLASH (B621)
A8E8 ...

A8B8 ZEB... ********** KEYWORD LOOKUP ***********************
A8BB NEXT ACCUMULATOR <A837>
A8BC COME POSSIBLE KEYWORDS IN TABLE
A8BF FOUND
A8E8 NO, "BAD AGAINST EACH (B66B)
A8F7 "IT" >>A927
A8EF ELSE IS IT ""? (FILE TYPE)
A8FC IF "CK THEN >>A8FC
A901 NO, "BAD KEYWORD >>A839
A906 ELSE "", IS IT PERMITTED ON THIS CMD?
A908 STR "ERROR >>A923
A910 "XARK WE HAVE "T" (B656)
A913 AND" "WITH TYPE INDEX OF 0 (B6CD)
A916 NOT LOCATE WHERE T VALUE IS TO GO (B6CE)
A918 IS IT "" 39 PARSE ONE CHAR <AA3A>
A91A YES "HERE?? >>A8F9
A91C IS IT A $7
A91E NO, "GAVE TYPE IN HEX >>A976
A920 ELSE IS "ALPHABETIC?"
A923 READY NEXT DECIMAL TYPE >>A960
A924 --- GO LOOKUP TYPE NAME IN TABLE >>A9B6
A926 "INH...
A927 "REX... VALID PARAMETER"
A929 GET... "V"
A92A IDG...
A92C IS "POSITION OF THIS KEYWORD (B975)
A92F "V" >>A947
A931 THIS KEYWORD PERMITTED? (B655)
A933 NOT WITH THIS COMMAND ANYWAY >>A923
A935 YES "D2
A93B YES "A941
A93A ELSE "ALREADY FOUND IT ON THIS LINE? (B657)
A941 MARK DON'T CHANGE DRIVE DEFAULT >>A947
A947 GEMS: ASSUME DRIVE = 1
A94F WE HAVE SLOW/DRIVE (B57)
A957 SIZE-1 IN BYTES OF VALUE (B97F)
<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
<th>NEXT OBJECT ADDR: A954</th>
</tr>
</thead>
<tbody>
<tr>
<td>A954</td>
<td>AND OFFSET TO VALUE IN STORAGE AREA</td>
<td></td>
</tr>
<tr>
<td>A957</td>
<td>PLUSH TO NON-BLANK &lt;A3A&gt;</td>
<td></td>
</tr>
<tr>
<td>A95A</td>
<td>NOTHING ELSE THERE? &gt;&gt;A9B0</td>
<td>(CAE)</td>
</tr>
<tr>
<td>A95C</td>
<td>IS NEXT CHAR A &quot;$&quot;?</td>
<td></td>
</tr>
<tr>
<td>A95E</td>
<td>YES, GO CONVERT HEX - ELSE, FALL THROUGH</td>
<td></td>
</tr>
</tbody>
</table>

**A960 *** CONVERT DECIMAL NUMBER ***** >>A976**

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
<th>NEXT OBJECT ADDR: A980</th>
</tr>
</thead>
<tbody>
<tr>
<td>A960</td>
<td>SAVE LINE INDEX (BE4B)</td>
<td></td>
</tr>
<tr>
<td>A963</td>
<td>CONVERT/ADD ONE DECIMAL DIGIT TO ACCUM</td>
<td></td>
</tr>
<tr>
<td>A966</td>
<td>OK. &gt;&gt;A96C</td>
<td></td>
</tr>
<tr>
<td>A968</td>
<td>OVERFLOW? THEN &quot;RANGE ERROR&quot; &gt;&gt;A9B3</td>
<td>&lt;&lt;AA5C&gt;</td>
</tr>
<tr>
<td>A96A</td>
<td>BAD DIGIT? THEN &quot;SYNTAX ERROR&quot; &gt;&gt;A9B0</td>
<td></td>
</tr>
<tr>
<td>A96C</td>
<td>RESTORE LINE INDEX (BE4B)</td>
<td></td>
</tr>
<tr>
<td>A96F</td>
<td>PLUSH TO NEXT NON-BLANK &lt;A3A&gt;</td>
<td></td>
</tr>
<tr>
<td>A972</td>
<td>AND GO BACK TO CONVERT NEXT DIGIT &gt;&gt;A984</td>
<td></td>
</tr>
<tr>
<td>A974</td>
<td>ALL DONE, END OF LINE OR COMMA &gt;&gt;A98F</td>
<td></td>
</tr>
</tbody>
</table>

**A976 *** CONVERT HEX NUMBER **********

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
<th>NEXT OBJECT ADDR: A980</th>
</tr>
</thead>
<tbody>
<tr>
<td>A976</td>
<td>PLUSH TO NEXT NON-BLANK (SKIP &quot;$&quot;)</td>
<td></td>
</tr>
<tr>
<td>A979</td>
<td>NOTHING LEFT? &gt;&gt;A9B8</td>
<td></td>
</tr>
<tr>
<td>A97B</td>
<td>SAVE LINE INDEX (BE4B)</td>
<td>(A3A)</td>
</tr>
<tr>
<td>A97E</td>
<td>CONVERT HEX DIGIT &lt;&lt;AAAA&gt;</td>
<td></td>
</tr>
<tr>
<td>A981</td>
<td>OK. &gt;&gt;A987</td>
<td></td>
</tr>
<tr>
<td>A983</td>
<td>OVERFLOW? THEN &quot;RANGE ERROR&quot; &gt;&gt;A9B3</td>
<td></td>
</tr>
<tr>
<td>A985</td>
<td>BAD DIGIT? THEN &quot;SYNTAX ERROR&quot; &gt;&gt;A9B0</td>
<td></td>
</tr>
<tr>
<td>A987</td>
<td>RESTORE LINE INDEX (BE4B)</td>
<td></td>
</tr>
<tr>
<td>A98A</td>
<td>PLUSH TO NEXT NON-BLANK &lt;A3A&gt;</td>
<td></td>
</tr>
<tr>
<td>A98D</td>
<td>AND GO CONVERT NEXT DIGIT &gt;&gt;A9B9</td>
<td></td>
</tr>
</tbody>
</table>

**A98F *** STORE KEYWORD VALUE **********

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
<th>NEXT OBJECT ADDR: A9B4</th>
</tr>
</thead>
<tbody>
<tr>
<td>A98F</td>
<td>HOW MANY BYTES TO CHECK?</td>
<td></td>
</tr>
<tr>
<td>A994</td>
<td>ALL HAVE BEEN CHECKED? &gt;&gt;A99E</td>
<td></td>
</tr>
<tr>
<td>A996</td>
<td>NO, INSURE MSB'S OF ACCUM ARE ZERO (85)</td>
<td></td>
</tr>
<tr>
<td>A999</td>
<td>IF NUMBER IS A SHORT INTEGER &gt;&gt;A9B3</td>
<td></td>
</tr>
<tr>
<td>A9A1</td>
<td>COPY ACCUM TO PROPER PARM STORAGE CFV'AF</td>
<td></td>
</tr>
<tr>
<td>A9A9</td>
<td>RESTORE LINE INDEX (BE4B)</td>
<td></td>
</tr>
<tr>
<td>A9AF</td>
<td>AND EXIT</td>
<td></td>
</tr>
<tr>
<td>A9B0</td>
<td>&quot;SYNTAX ERROR&quot; JUMP &gt;&gt;A839</td>
<td></td>
</tr>
<tr>
<td>A9B3</td>
<td>&quot;RANGE ERROR&quot; JUMP &gt;&gt;A75E</td>
<td></td>
</tr>
</tbody>
</table>

**A9B6 *** STORE KEYWORD VALUE **********

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
<th>NEXT OBJECT ADDR: A9B4</th>
</tr>
</thead>
<tbody>
<tr>
<td>A9B6</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>A9B8</td>
<td>COPY 3 CHARACTER TYPE TO ACCUM (BCAF)</td>
<td></td>
</tr>
<tr>
<td>A9B8</td>
<td>(COPIED ALL 37) &gt;&gt;A9C7</td>
<td></td>
</tr>
<tr>
<td>A9C8</td>
<td>(GET NEXT CHAR IGNORING BLANKS) &lt;&lt;A9A3</td>
<td></td>
</tr>
<tr>
<td>A9C5</td>
<td>MUST HAVE 3 CHARACTERS &gt;&gt;A9B0</td>
<td></td>
</tr>
</tbody>
</table>
**BASIC Interpreter (B1) -- V1.1.1 -- 18 JUN 84**

**ADDR DESCRIPTION**

**AA3A *************** FLUSH TO NON-BLANK ***********************
Z-FLAG
SET IF COMMA OR RETURN FOUND
C-FLAG
SET IF COMMA

AA3A IGNORE BLANKS
AA3F GET NEXT NON-BLANK <AA4A>
AA42 COMMA?
AA44 YES, OUT >>AA49
AA46 RETURN?
AA48 EXIT INDICATING WHAT WE FOUND
AA49 RETURN

**AA4A *************** GET NEXT CHARACTER ***********************
AA4A GET NEXT CHAR IN INPUT LINE (0200)
AA4D FORCE OFF MSB
AA4F LOWER CASE?
AA51 NO >>AA55
AA53 YES, FORCE UPPER CASE
AA55 BUMP LINE INDEX
AA56 IS THIS A FLUSH CHARACTER (LIKE BLANK)? (BCA9)
AA59 YES, GO GET NEXT ONE >>AA4A
AA5B ELSE, RETURN WITH IT

**AA5C *************** CONVERT DIGIT AND ADD TO ACCUM ***********************
AA5C NUMERIC?
AA5E NO >>AA64
AA62 YES >>AA65
AA64 NOT NUMERIC, EXIT WITH CARRY SET
AA65 AND Z-FLAG RESET
AA67 RETURN
AA68 ISOLATE DECIMAL PORTION OF DIGIT
AA6B CURRENT VALUE OF ACCUM... (BCB1)
AA6E >1,703,932
AA70 YES, OVERFLOW >>AA94
AA74 PUSH ENTIRE ACCUM ONTO STACK (BCA9)
AA77 ACCUM*2 (DO IT ONCE) <AA97>
AA7E ACCUM*4 (AND AGAIN) <AA7F>
AA84 ----
AA85 ACCUM*4+ACCUM --> ACCUM*5 (BCA9)
AA91 FINALLY, ACCUM*5*2 --> ACCUM*10 <AA7F>
AA94 ----
AA95 ACCUM OVERFLOW? >>AAA
AA97 NO, ADD NEW DIGIT TO ACCUM (BCA9)
AA9A AND STORE IT (BCA9)
AA9D NO CARRY? >>AA9D
AA90 GOT CARRY, PROPAGATE IT THRU ACCUM (BCB9)
AAA OVERFLOW ERROR
AAD NORMAL EXIT

**AAEA *************** CONVERT HEX DIGIT AND ADD ***********************
AAAE NUMERIC?
AAAB NO >>AAAE
AAAE YES >>AAAC4
AAAC4 NON-NUMERIC, HOW BOUT "A" THRU
AAAC5 "F"
AAAC6 YES1 >>AAAC7
AAAC7 ---
AAAC8 NO, GET OUT NOW
AAAC9 RETURN
AAACB "A" THRU "F", CONVERT TO $8A-$BF
AAAC4 ISOLATE DIGIT
AAAC8 SHIFT ACCUM 4 BITS LEFT TO MAKE ROOM <AAAD7>
AAACB (WATCH OUT FOR OVERFLOW) >>AAEA
AAAD6 OR IN NEW NIBBLE (BCA9)
AAAD7 AND REPLACE IN ACCUM LSB (BCA9)
AAAD8 DONE

**AAAD7 *************** SHIFT 3 BYTE ACCUM LEFT A BIT ***********************
AAAD7 SHIFT THE THREE BYTE WORK ACCUM (BCA9)
AAAE0 RETURN

**AAE1 *************** SCAN CMD TABLE FOR COMMAND ***********************
AAE1 START WITH LAST COMMAND IN TABLE
AAE6 IS IT A "=" COMMAND? (BCB0)
AAE8 NOPE >>AAE5
AAE5 YES, SPECIAL COMMAND NUMBER (BE53)
AAE9 ZERO LENGTH COMMAND STRING (BE52)
AAF3 CONTINUE >>AA12
AAE5 FIRST COMMANDS IN TABLE ARE 8 CHARS
AAFA GET INDEX TO NAME --> (BE58)
AAFD SAME LENGTH AS LAST NAME? >>AAE5
AAF0 NO,
AAF2 NAMES ARE ONE BYTE SHORTER FROM NOW ON (BE52)
AAF5 COMPAIR HIS NAME TO MY TABLE (BCB0)
AAF0C NOT IT... >>AAE5
AAF18 COMPARE ENTIRE NAME >>AAE5
AAF12 FOUND IT! GET COMMAND INDEX (BE53)
AAF15 "#" FOR MOST THINGS
AAE17 Nail UP PERMITTED PARMS BITS (B92A)
AAE23 EXIT HAPPILY
AAE24 RETURN
**BASIC Interpreter (BI) -- V1.1.1 -- 18 JUN 84**
NEXT OBJECT ADDR: AB24

**DESCRIPTION/CONTENTS**

**ADDR**

```
**THE ONE, SKIP TO NEXT (BE52)**
AB25 BE52
AB26 BE52
AB27 BE52
AB28 BE52
AB29 BE52
AB30 BE52
AB31 BE52
AB32 BE52
AB33 BE52
AB34 BE52
AB35 BE52
AB36 BE52
AB37 BE52
AB38 BE52
AB39 BE52
AB40 BE52
AB41 BE52
AB42 BE52
AB43 BE52
AB44 BE52
AB45 BE52
AB46 BE52
AB47 BE52
AB48 BE52
AB49 BE52
AB50 BE52
AB51 BE52
AB52 BE52
AB53 BE52
AB54 BE52
AB55 BE52
AB56 BE52
AB57 BE52
AB58 BE52
AB59 BE52
AB60 BE52
AB61 BE52
AB62 BE52
AB63 BE52
AB64 BE52
AB65 BE52
AB66 BE52
AB67 BE52
AB68 BE52
AB69 BE52
AB70 BE52
AB71 BE52
AB72 BE52
AB73 BE52
AB74 BE52
AB75 BE52
AB76 BE52
AB77 BE52
AB78 BE52
AB79 BE52
AB80 BE52
AB81 BE52
AB82 BE52
AB83 BE52
AB84 BE52
AB85 BE52
AB86 BE52
AB87 BE52
AB88 BE52
AB89 BE52
AB90 BE52
AB91 BE52
AB92 BE52
AB93 BE52
AB94 BE52
AB95 BE52
AB96 BE52
AB97 BE52
AB98 BE52
AB99 BE52
AB100 BE52
```

**ABB2 ********** "RUN" COMMAND ***************

ABB2 NO INPUT FILE ACTIVE NOW
ABB7 NO APPLESOF ERROR NUMBER
ABB8 CLEAR PATHNAME1
ABB5 NO ERROR >>ABB5
ABBF YES, LOAD PROGRAM <<ABBF
ABB2 ERROR? >>AC14
ABB4 NO, CLEAR VARIABLES <<D655
ABB7 CLEAR ERROR FLAG
ABB9 POSITION TO LINE NUMBER IF GIVEN <<AC97
ABB5 RESTORE MY INTERCEPTS <<ABB5
ABB3 CLEAR COMMAND NUMBER ETC, MODE = 4 <<ABB5
ABB2 JUMP INTO APPLESOF TO RUN PROGRAM >>D7D2

**ABB5 ********** CLEAR COMMAND NUMBER ETC. ***************

ABB5 SET NORMAL (NON-INVERSE OR FLASH) <<P37
ABB4 SEARCH CHARACTER FOR TRACE IS "#" <<P661
ABB5 NO COMMAND NUMBER NOW (BE55)
ABB2 NO PROMPT
ABB6 SET MODE=4 (DEFERRED) <<P77
ABB7 "SYNTAX ERROR" IF THINGS GO WRONG >>ABB9

**ABB7 ********** "LOAD" COMMAND ***************

ABB7 LOAD PROGRAM <<ABB7
ABBF ERROR? IF NOT, FALLTHRU TO WARMSTART >>AC14

**ABB1 ********** WARMDO$: WARMSTART BI ***************

ABB1 CLEAR APPLESOF, RESET POINTERS <<D655
ABB4 SET MODE=SET INTERCEPTS <<A17
ABB9 CURSOR HORIZ. = 0 (START OF LINE)
ABB9 GO WARMSTART APPLESOF >>D43F

**ABBF ********** LOAD A PROGRAM ***************

ABBF CLOSE ALL OPEN FILES <<BBF
AC90 ERROR? >>AC14
AC90 ALL HIDDEN FILES <<AC90
AC95 GO LOAD FILE <<AC15
AC96 ERROR? >>AC14
AC99 SET LOMEM = ARRAYS = FREESTART
AC14 ALL TO END OF PROGRAM LOADED
AC14 RETURN
**BASIC Interpreter (Bl) -- V1.1.1 -- 18 JUN 84**
**NEXT OBJECT ADDR: AC15**

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>AC15</td>
<td>READ A PROGRAM FROM A FILE *********************************</td>
</tr>
<tr>
<td>AC17</td>
<td>READ REQUESTED</td>
</tr>
<tr>
<td>AC19</td>
<td>OPEN THE FILE &lt;B194&gt;</td>
</tr>
<tr>
<td>AC1C</td>
<td>ERROR? &gt;&gt;AC14</td>
</tr>
<tr>
<td>AC20</td>
<td>MLI: GET EGF &lt;BE70&gt;</td>
</tr>
<tr>
<td>AC23</td>
<td>ERROR? &gt;&gt;AC14</td>
</tr>
<tr>
<td>AC27</td>
<td>APPLESOFT PROGRAM START --&gt; READ DATA (BED7)</td>
</tr>
<tr>
<td>AC2A</td>
<td>ADD TO THAT THE EOF MARK TO ... (BEC8)</td>
</tr>
<tr>
<td>AC2D</td>
<td>SET AD PARM --&gt; END OF PROGRAM IMAGE (BE58)</td>
</tr>
<tr>
<td>AC3B</td>
<td>OVERFLOW? &gt;&gt;AC3F</td>
</tr>
<tr>
<td>AC3D</td>
<td>NO, WOULD PROGRAM EXCEED HIMEM?</td>
</tr>
<tr>
<td>AC3F</td>
<td>IF SO...</td>
</tr>
<tr>
<td>AC41</td>
<td>&quot;PROGRAM TOO LARGE&quot; &gt;&gt;AC14</td>
</tr>
<tr>
<td>AC43</td>
<td>ELSE, PICK UP LENGTH AGAIN (BEC8)</td>
</tr>
<tr>
<td>AC49</td>
<td>AND GO READ IT IN &lt;AF98&gt;</td>
</tr>
<tr>
<td>AC4C</td>
<td>ERROR? &gt;&gt;AC14</td>
</tr>
<tr>
<td>AC4E</td>
<td>CLOSE FILE &lt;AF94&gt;</td>
</tr>
<tr>
<td>AC51</td>
<td>ERROR? &gt;&gt;AC14</td>
</tr>
<tr>
<td>AC53</td>
<td>RELOCATE PROGRAM IF NECESSARY &lt;AC61&gt;</td>
</tr>
<tr>
<td>AC5C</td>
<td>COPY AD PARM TO APPLESOFT PGM END PTR</td>
</tr>
<tr>
<td>AC5E</td>
<td>RETURN</td>
</tr>
</tbody>
</table>

**AC61** RELOCATE APPLESOFT PROGRAM **************************

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>AC61</td>
<td></td>
</tr>
<tr>
<td>AC62</td>
<td>RKS APPLESOFT PROGRAM SAVED FROM SAME</td>
</tr>
<tr>
<td>AC64</td>
<td>MEMORY LOCATION? (BE99)</td>
</tr>
<tr>
<td>AC73</td>
<td>YES, NOTHING TO DO THEN &gt;&gt;ACBA</td>
</tr>
<tr>
<td>AC79</td>
<td>ELSE, LOOP THROUGH PROGRAM</td>
</tr>
<tr>
<td>AC7B</td>
<td>ADJUSTING ALL ADDRESSES TO</td>
</tr>
<tr>
<td>AC7D</td>
<td>THE NEW LOAD LOCATION</td>
</tr>
</tbody>
</table>

**AC97** POSITION TO LINE NUMBER *****************************

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>AC97</td>
<td>WAS A LINE NUMBER PARM GIVEN? (BE57)</td>
</tr>
<tr>
<td>AC9D</td>
<td>NO, NEVER MIND &gt;&gt;ACBA</td>
</tr>
<tr>
<td>AC9F</td>
<td>COPY L KEYWORD VALUE TO APPLESOFT'S LINE # (BE68)</td>
</tr>
<tr>
<td>ACA9</td>
<td>THEN CALL APPLESOFT TO FIND THE LINE &lt;D61A&gt;</td>
</tr>
<tr>
<td>ACAP</td>
<td>SUBTRACT ONE FROM THE ADDRESS</td>
</tr>
<tr>
<td>ACB1</td>
<td>AND POINT APPLESOFT'S GETCHR SUBROUTINE</td>
</tr>
<tr>
<td>ACB3</td>
<td>AT IT (SO NEXT CHAR READ WILL BE FIRST</td>
</tr>
<tr>
<td>ACB5</td>
<td>CHARACTER ON THE LINE).</td>
</tr>
<tr>
<td>ACBA</td>
<td>RETURN</td>
</tr>
</tbody>
</table>

**ACBB** "SAVE" COMMAND ***************************

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACBB</td>
<td>DOES FILE EXIST ALREADY? &gt;&gt;ACDF</td>
</tr>
<tr>
<td>ACBD</td>
<td>NO, TYPE = BAS</td>
</tr>
<tr>
<td>ACBF</td>
<td>IN T KEYWORD VALUE (BE6A)</td>
</tr>
<tr>
<td>ACC2</td>
<td>AND MLI LIST (BE88)</td>
</tr>
<tr>
<td>ACC7</td>
<td>ALLOW ALL ACCESSES (READ/WRITE/ETC.) (BE87)</td>
</tr>
<tr>
<td>ACC8</td>
<td>SAVE PROGRAM START ADDRESS IN (BEA5)</td>
</tr>
<tr>
<td>ACCF</td>
<td>AUXID'S (BE99)</td>
</tr>
<tr>
<td>ACD0</td>
<td>GO CREATE A NEW FILE &lt;AD46&gt;</td>
</tr>
<tr>
<td>ACDD</td>
<td>ERROR? &gt;&gt;AD28</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACDF</td>
<td>WRITE ACCESS REQUESTED</td>
</tr>
<tr>
<td>ACE1</td>
<td>BAS TYPE FILE</td>
</tr>
<tr>
<td>ACE3</td>
<td>OPEN IT &lt;B194&gt;</td>
</tr>
<tr>
<td>ACE6</td>
<td>ERROR? &gt;&gt;AD28</td>
</tr>
<tr>
<td>ACEB</td>
<td>SUBTRACT APPLESOFT PTRS TO COMPUTE</td>
</tr>
<tr>
<td>ACED</td>
<td>LENGTH OF PROGRAM.</td>
</tr>
<tr>
<td>ACEE</td>
<td>STORE THIS IN EOF MARK LIST (BEC8)</td>
</tr>
<tr>
<td>ACFB</td>
<td>MBG OF EOF MARK IS 80 (&lt;&lt;4K PGM) (BEC4)</td>
</tr>
<tr>
<td>ACG0</td>
<td>POINT LIST TO PROGRAM AS DATA TO WRITE (BED7)</td>
</tr>
<tr>
<td>ACG8</td>
<td>WRITE A RANGE TO DISK FILE &lt;AF9C&gt;</td>
</tr>
<tr>
<td>ACGD</td>
<td>ERROR? &gt;&gt;AD28</td>
</tr>
<tr>
<td>ACGF</td>
<td>MLI: SET EOF (TO TRUNCATE OLD LONGER FILE) &lt;BE70&gt;</td>
</tr>
<tr>
<td>ACGJ</td>
<td>ERROR? &gt;&gt;AD28</td>
</tr>
<tr>
<td>ACGK</td>
<td>CLOSE THE FILE &lt;AF94&gt;</td>
</tr>
<tr>
<td>ADC0</td>
<td>NO, CHANGE IT &gt;&gt;AD29</td>
</tr>
<tr>
<td>AD28</td>
<td>ELSE, EXIT</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>AD29</td>
<td>TO CHANGE IT, (BE99)</td>
</tr>
<tr>
<td>AD2F</td>
<td>EXIT THRU SET FILE INFO ROUTINE &gt;&gt;B7D9</td>
</tr>
</tbody>
</table>

**AD32** "CREATE" COMMAND ******************************

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>AD32</td>
<td>AUXID = 0 (A$ OR RECLN)</td>
</tr>
<tr>
<td>AD3D</td>
<td>TYPE KEYWORD GIVEN?</td>
</tr>
<tr>
<td>AD3F</td>
<td>YES &gt;&gt;AD46</td>
</tr>
<tr>
<td>AD43</td>
<td>NO, ASSUME TYPE = DIR (BE6A)</td>
</tr>
<tr>
<td>AD46</td>
<td>CREATE FILE ENTRY *** (BE43)</td>
</tr>
<tr>
<td>AD49</td>
<td>EXEC FILE ACTIVE?</td>
</tr>
<tr>
<td>AD4C</td>
<td>HOW MANY FILES ARE OPEN INCLUDING EXEC? (BE4D)</td>
</tr>
<tr>
<td>AD4F</td>
<td>8 OR MORE?</td>
</tr>
<tr>
<td>AD51</td>
<td>YES, ERROR &gt;&gt;AD6E</td>
</tr>
<tr>
<td>AD56</td>
<td>ELSE, SET TYPE IN MLI LIST (BEA4)</td>
</tr>
<tr>
<td>AD59</td>
<td>FULL ACCESS (READ/WRITE/ETC.)</td>
</tr>
<tr>
<td>AD5B</td>
<td>KIND = STANDARD FILE</td>
</tr>
<tr>
<td>AD5D</td>
<td>DIR FILE WANTED?</td>
</tr>
</tbody>
</table>
BASIC Interpreter (BI) -- V1.1.1 -- 18 JUN 84  NEXT OBJECT ADDR: AD5F

AD5F  WRH > AD63
AD61  D65, KIND = DIR FILE
AD63  SET ACCESS (B6A3)
AD66  NO, KIND = (B6A7)
AD68  NO! CREATE (DON'T COME BACK HERE) > > B78
AD6E  "RAM TOO LARGE" ERROR
AD70  RETURN

AD71  ********** "RENAME" COMMAND ***************

AD71  ---
AD75  SECOND PATHNAME GIVEN?
AD78  IF SO, GO ML1: RENAME > AD7F
AD7A  SYNTAX ERROR OTHERWISE > A389

AD7D  ********** "DELETE" COMMAND ********************

AD7D  ML1: DELETE CALL TYPE
AD7F  EXIT THRU ML1 CALL > B78

AD82  ********** "LOCK" COMMAND ********************

AD82  GET FILE INFO FOR PATHNAME < B7D0>
AD85  GET ACCESS CODES (B6B7)
AD88  TOP OFF ALL...
AD8A  BUT READ
AD8F  THEN GO SET UPDATED FILE INFO > B7E7

AD92  ********** "UNLOCK" COMMAND ********************

AD92  GET FILE INFO FOR PATHNAME < B7D0>
AD95  TOP ON ALL FILE ACCESSES
AD9D  THEN GO SET UPDATED FILE INFO > B7E7

AD98  ********** "PREFIX" COMMAND ********************

AD98  SKIP DRIVE GIVEN ON COMMAND? (B657)
AD9A  IF SO, GOT OPEN AND ALREADY > ADAC
AD9B  ELSE, (B656)
AD9C  DO NOT FOR PATHNAME
ADAC  ML1: SET PREFIX...
ADAE  IF IT'S THERE > AD7F
ADBB  ELSE, IS BASIC PROGRAM RUNNING?
ADBC  IF SO, SET PREFIX ACTIVE FLAG > ADD1
ADBD  NEW LINE <9FA6>
ADBC  NAME OF NAME YET? > ADCA
ADBD  NO, COPY NAME IN PATHNAME BUFABER (BCA6)
ADDC  TO OUTPUT DEVICE <9FA6>
ADDE  AND SKIP A BLANK LINE <9FAB>
ADDF  DONE

AD23  **************** "BLOAD" COMMAND *******

AD23  READING...
AD25  TYPE = BIN
AD27  OPEN THE FILE < B194>
AD2A  ERROR? > AD14
AD2C  ASSUME USER SPECIFIED AD KEYWORD (B658)
AD35  IF SO, USE HIS ADDRESS > AE47
AD37  ELSE, USE AD IN FILE INFO AUXID (B6B9)
AD40  WAS T KEYWORD GIVEN?
AD42  YES, INVALID FROM (ONLY BIN IS LEGAL) > AE78
AD47  POINT READ/WRITE FARMS TO DATA (B6D7)
BASIC Interpreter (BI) -- V1.1.1 -- 18 JUN 84  NEXT OBJECT ADDR: AED4
ADR DESCRIPTION/CONTENTS
-----------------------------------------------
AE40 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
AE8E YES, "PROGRAM TOO LARGE"
AE90 ---
AE91 RETURN
AE92 STORE LENGTH TO READ OR WRITE (BED9)
AE9B B KEYWORD GIVEN?
AE9D NO >>AEC4
AEAB YES, COPY B VALUE TO SET MARK LIST (BE5A)
AEAC ---
AEAC MLI: SET MARK <BE78>
AEB2 NO ERROR? >>AEC4
AEB4 ERROR, RANGE ERROR?
AEB6 NO >>AE90
AEB8 BSAVING (NOT BLOAD/BRUNING)?
AEBA NO >>AE90
AEBC MLI: FORCE EOF FORWARD TO MARK <BE70>
ACE1 AND TRY SET MARK AGAIN >>AEAA
ACE3 RETURN
ACE4 GET COMMAND NUMBER (BE53)
ACE7 ASSUME READ
ACE9 BSAVE?
ACEB NO, READ IS CORRECT >>AECF
ACED WRITING
ACEF 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: AED4
ADR DESCRIPTION/CONTENTS
-----------------------------------------------
AED7 ********** "STORE" COMMAND **********************
AED7 PATHNAME1 EXISTS? >>AEEB
AED9 NO, T = VAR BY DEFAULT
AEE1 FULL ACCESS (READ/WRITE/ETC.)
AEE6 CREATE THE FILE <AD46>
AEE9 ERROR? >>AF39
AEEB COMPRESS APPLESOFT VARS AGAINST HIMEM <A40D>
AEE4 OPEN "VAR" FILE FOR WRITE <B194>
AEE7 ERROR? >>AF32
AEE9 POINT TO INTERNAL 5 BYTE HEADER BUFFER <AF3A>
AEEC AND WRITE OUT LENGTHS OF VARS <AF9C>
AEFF ERROR? >>AF32
AF01 STORE ADDRESS OF VARS (BC8E)
AF04 IN READ/WRITE PARM LIST (BED7)
AF07 AND FILE INFO AUXID (BB9)
AF13 GET LENGTH OF VARS (BC91)
AF19 AND WRITE THEM OUT <AF9C>
AF1C ERROR? >>AF32
AF20 MLI: GET MARK <BE70>
AF25 MLI: SET NEW EOF (TRUNCATE IF NECESSARY) <BE70>
AF28 ERROR? >>AF32
AF2A MLI: SET FILE INFO WITH AD OF VARS <B7D9>
AF2D ERROR? >>AF32
AF2F CLOSE FILE <AF94>
AF32 ---
AF34 REEXPAND VARS BACK AGAIN <A472>
AF39 RETURN

AP3A ********** SETUP TO READ/WRITE VAR HDR **************
AP3A APPLESOFT VARIABLES HEADER CONSISTS OF:
AP3A 2 BYTE LENGTH OF SIMPLE+ARRAY VARIABLES
AP3A 2 BYTE LENGTH OF SIMPLE VARIABLES ONLY
AP3A 1 BYTE MSB OF HIMEM FOR THESE VARIABLES
AP3A STORE ADDRESS OF 5 BYTE INFO
AP3C IN READ/WRITE PARM LIST (BED7)
AP46 LENGTH = 5
AP48 RETURN

AP49 ********** "RESTORE" COMMAND **************
AP49 TYPE = VAR
AP4B READING
AP4D OPEN THE FILE <B194>
AP50 ERROR? >>AF39
AP52 SET UP TO READ THE HEADER <AF3A>
AP55 READ 5 BYTE HEADER <AF9B>
AP58 ERROR? >>AF39
AP5A PICK UP WHERE TO READ IN COMPRESSED VARS (BE9B)
**Beneath DESCRIPTION/CONTENTS**

```plaintext
POM AUXID (BC6E)

BASIC IN THIS PROGRAM INVESTIGATE THIS BY THE DIFFERENCE
BETWEEN HMEM'S (NOW AND WHEN STORED) (BC8D)

APRIVATE VAR'S WON'T OVERLAY PROGRAM

SO, ERROR >>APF9

COMPUTE LENGTH OF ALL VAR'S/STRINGS

HMEM-START (BC8F)

ALEAD COMBINED VAR'S INTO MEMORY <AF98>

APF6 ERROR >>APF9

APF7 MOSE THE FILE >>APF9

APF8 EXIT BY REEXPANDING THE VAR'S DOWN >>APF2

APF9 PROGRAM TOO LARGE" ERROR

APF3 GOTO

APF3 ER

APF3A ** CLOSE FILE ****************************************

APF3D RW

APF3F T MLI CLOSE OPCODE

APF9 GOTO TO MLI >>APF3A

APF9A **

APF9B READ/WRITE A RANGE ***********************************

APF9C READ MLI OPCODE

APF9D SWIP IN >>APF9E

APF9F WRITE MLI OPCODE

APF9G STORE LENGTH (BEDA)

APF9H GOTO THRU MLI:READ OR WRITE >>BE70

APF9I ** "PR#' COMMAND ******************************

APF9J P

APF9K WHERE CSWL AND OUTVEC

APF9L SWIP TO COMMON CODE >>APF5

APF9M P

APF9N ** "IN#' COMMAND ******************************

APF9O KSWL

APF9P ID IN INVE

APF9Q P

APF9A IN SLOT GIVEN BY USER (BE68)

APF9E FOR USE AS INDEX INTO TABLE

APF9F IN SLOT PARAMETER GIVEN?

APF9G I... >>APD2

APF9H ANS, (BE57)

APF9I GIVEN? >>APF7

APF9J GET INVEO OR OUTVEC FOR THIS SLOT (BE10)

APF9K STORE ON AD KEYWORD VALUE (BE58)

APF9L VALIDITY CHECK I/O DRIVER <<AF9>

APF9M GOOD? >>APF6

APF9N CALL INVEC OR PKWL (CA9)

APF9O AND REPLACE ONE OR THE OTHER WITH (BE59)

APF9A A

APF9B V

APF9C N

APF9D A

APF9E A

BASIC DESCRIPTION/CONTENTS

APF6 RETURN

APF7 VALIDITY CHECK AD KEYWORD VALUE <AF9>

APF8 IS DRIVER IN MAIN RAM (BE58)

APF9 GOOD, COPY VALUE TO INVEO OR OUTVEC (BE59)

APF9 EXIT BUT DON'T REDIRECT I/O NOW

APF9G ** VALIDITY CHECK I/O DRIVER *********************

APF9H $3A/3B --> NEW HANDLER (FROM AD PARM) (BE58)

APF9I IS DRIVER IN MAIN RAM (BE59)

APF9J TEST ROM AT USER'S ADDRESS

APF9K FOR STABILITY

APF9L TIMES

APF9M MUST BE OK

APF9N RETURN

APF9O MAIN RAM I/O DRIVER

APF9P MUST START WITH A "CLD" INSTRUCTION

APF9Q OK... >>B01C

APF9R ELSE, "NO DEVICE CONNECTED"

APF9S RETURN

APF9T "BYE" COMMAND ************************************

APF9U CLOSE ANY OPEN FILES <<B4F2>

APF9V CLOSE EXEC FILE, IF ANY <<B2F2>

APF9W MLI CALL: <<B0F0>

APF9X EXIT

APF9Y USE READ PARM LIST BECAUSE QUIT DOESN'T NEED PARM.

APF9Z ** "CAT" COMMAND *****************************

APF9A 9 CHARACTERS PER LINE

APF9B THEN PROCESS LIKE "CATALOG" >>B03C

APF9C ** "CATALOG" COMMAND **************************

APF9D 9 CHARACTERS PER LINE

APF9E STORE LINE LENGTH (BCB6)

APF9F TEST FOR T AND

APF9G "PATHNAME" GIVEN

APF9H $T >>B04A

APF9I NO T, "T=0 (ANY TYPE WILL DO) (BE6A)

APF9J GOT PATHNAME >>B055

APF9K NG PATHNAME, GET FILE INFO FOR PREFIX <BE70>

APF9L ERROR >>B007

B051 OPEN/READ DIRECTORY HEADER <B14A>
```
BASIC Interpreter (B1) -- V1.1.1 -- 18 JUN 84 NEXT OBJECT ADDR: B054

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>B054</td>
<td>ERROR? &gt;&gt; B087</td>
</tr>
<tr>
<td>B056</td>
<td>SKIP TO A NEW LINE</td>
</tr>
<tr>
<td>B059</td>
<td>FORMAT DIRECTORY?</td>
</tr>
<tr>
<td>B05C</td>
<td>PRINT $201 &lt;9F9D&gt;</td>
</tr>
<tr>
<td>B05F</td>
<td>SKIP TO A NEW LINE</td>
</tr>
<tr>
<td>B062</td>
<td>BLANK $201 BUFFER</td>
</tr>
<tr>
<td>B067</td>
<td>UNPACK HEADER M$</td>
</tr>
<tr>
<td>B06A</td>
<td>PRINT IT (40 OR 80)</td>
</tr>
<tr>
<td>B06D</td>
<td>SKIP TO A NEW LINE</td>
</tr>
<tr>
<td>B073</td>
<td>ANY FILES IN THIS</td>
</tr>
<tr>
<td>B076</td>
<td>NO &gt;&gt; B0A3</td>
</tr>
<tr>
<td>B078</td>
<td>YES, READ NEXT ENT.</td>
</tr>
<tr>
<td>B07B</td>
<td>ERROR? &gt;&gt; B0B7</td>
</tr>
<tr>
<td>B07D</td>
<td>GET TYPE REQUESTED</td>
</tr>
<tr>
<td>B083</td>
<td>ANY TYPE WILL DO?</td>
</tr>
<tr>
<td>B082</td>
<td>NO, CHECK TYPE AG</td>
</tr>
<tr>
<td>B085</td>
<td>NOT IT, SKIP IT &gt;&gt;</td>
</tr>
<tr>
<td>B087</td>
<td>ELSE, FORMAT ENTRY</td>
</tr>
<tr>
<td>B09A</td>
<td>AND PRINT $201 &lt;B80D</td>
</tr>
<tr>
<td>B09D</td>
<td>CHECK KEYBOARD (C0</td>
</tr>
<tr>
<td>B09F</td>
<td>FOR A CONTROL-C</td>
</tr>
<tr>
<td>B092</td>
<td>IGNORE ANYTHING ELSE</td>
</tr>
<tr>
<td>B094</td>
<td>CONTROL-C, WHAT IS</td>
</tr>
<tr>
<td>B097</td>
<td>DEFERRED &gt;&gt; B0A3</td>
</tr>
<tr>
<td>B099</td>
<td>NO, IMMEDIATE, RET</td>
</tr>
<tr>
<td>B09C</td>
<td>AND EXIT RIGHT NOW</td>
</tr>
<tr>
<td>B09E</td>
<td>ELSE, ANY FILES IN?</td>
</tr>
<tr>
<td>B0A1</td>
<td>YES, CONTINUE &gt;&gt; B0B3</td>
</tr>
<tr>
<td>B0A3</td>
<td>ELSE, CLOSE DIRECTORY</td>
</tr>
<tr>
<td>B0A6</td>
<td>ERROR? &gt;&gt; B0B7</td>
</tr>
<tr>
<td>B0A8</td>
<td>SKIP TO A NEW LINE</td>
</tr>
<tr>
<td>B0AB</td>
<td>FORMAT BLOCKS FREE</td>
</tr>
<tr>
<td>B0AE</td>
<td>ERROR? &gt;&gt; B0B7</td>
</tr>
<tr>
<td>B0B0</td>
<td>PRINT $201 &lt;9F9D&gt;</td>
</tr>
<tr>
<td>B0B3</td>
<td>SKIP A LINE &lt;9FAB&gt;</td>
</tr>
<tr>
<td>B0B7</td>
<td>DONE</td>
</tr>
</tbody>
</table>

B0B8 ******* FORMAT IN

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>B0B8</td>
<td>BLANK $201 BUFFER</td>
</tr>
<tr>
<td>B0B8</td>
<td>FILE NAME IS AT</td>
</tr>
<tr>
<td>B0BD</td>
<td>NAME OF DIRECTORY</td>
</tr>
<tr>
<td>B0C2</td>
<td>VOLUME DIRECTOY TO</td>
</tr>
<tr>
<td>B0C4</td>
<td>NO &gt;&gt; B0CC</td>
</tr>
<tr>
<td>B0C6</td>
<td>YES, START NAME</td>
</tr>
<tr>
<td>B0CA</td>
<td>---</td>
</tr>
<tr>
<td>B0CB</td>
<td>ISOLATE NAME LENGTH</td>
</tr>
<tr>
<td>B0CD</td>
<td>AND SET UP LENGTH?</td>
</tr>
<tr>
<td>B0D2</td>
<td>COPY DIRECTORY NAME</td>
</tr>
</tbody>
</table>

B0E7 ******* FORMAT BLOCKS FREE/INUSE

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>B0E7</td>
<td>POINT ML1:ONLINE PARMLIST</td>
</tr>
<tr>
<td>B0E9</td>
<td>TO TXTBUF (PATHNAME) (BEC8)</td>
</tr>
<tr>
<td>B0F1</td>
<td>COPY DEVICE NUMBER (UNIT) (BFB3)</td>
</tr>
<tr>
<td>B0F9</td>
<td>ML1: ONLINE &lt;BE70&gt;</td>
</tr>
<tr>
<td>B0FC</td>
<td>ERROR? &gt;&gt; B0B7</td>
</tr>
<tr>
<td>B101</td>
<td>ISOLATE NAME LENGTH FROM BUFFER</td>
</tr>
<tr>
<td>B104</td>
<td>BUMP BY ONE TO INCLUDE &quot;/&quot;</td>
</tr>
<tr>
<td>B105</td>
<td>AND STORE IT AS A PREFIX (BCBC)</td>
</tr>
<tr>
<td>B10A</td>
<td>STORE &quot;/&quot; AS FIRST CHARACTER (BCBD)</td>
</tr>
<tr>
<td>B10D</td>
<td>GET FILE INFO FOR PREFIX &lt;B70&gt;</td>
</tr>
<tr>
<td>B110</td>
<td>ERROR? &gt;&gt; B0B7</td>
</tr>
<tr>
<td>B112</td>
<td>BLANK $201 BUFFER &lt;A6C4&gt;</td>
</tr>
<tr>
<td>B117</td>
<td>UNPACK &quot;BLOCKS FREE; BLOCKS USED...&quot; &lt;9F90&gt;</td>
</tr>
<tr>
<td>B11A</td>
<td>ZERO THE THREE BYTE ACCUM &lt;A837&gt;</td>
</tr>
<tr>
<td>B125</td>
<td>CONVERT AUXID (TOTAL BLOCKS) &lt;A62F&gt;</td>
</tr>
<tr>
<td>B130</td>
<td>CONVER BLOCKS USED &lt;A62F&gt;</td>
</tr>
<tr>
<td>B137</td>
<td>BLOCKS_FREE = TOTAL_BLOCKS &lt;BEC8&gt;</td>
</tr>
<tr>
<td>B13E</td>
<td>... = BLOCKS USED (BBED)</td>
</tr>
<tr>
<td>B145</td>
<td>CONV BLOCKS FREE &lt;A62F&gt;</td>
</tr>
<tr>
<td>B149</td>
<td>DONE1</td>
</tr>
</tbody>
</table>

B14A ******* OPEN/READ DIRECTORY HDR

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>B14A</td>
<td>READ ONLY</td>
</tr>
<tr>
<td>B14E</td>
<td>CHECK FILE KIND (BBEB)</td>
</tr>
<tr>
<td>B151</td>
<td>VOLUME DIRECTOY?</td>
</tr>
<tr>
<td>B152</td>
<td>NO &gt;&gt; B158</td>
</tr>
<tr>
<td>B153</td>
<td>YES, TYPE = DIR (BBEB)</td>
</tr>
<tr>
<td>B158</td>
<td>OPEN THE FILE &lt;B1A0&gt;</td>
</tr>
<tr>
<td>B15B</td>
<td>ERROR? IF NOT, FALL THRU &gt;&gt; B193</td>
</tr>
</tbody>
</table>

B15D ******* READ DIRECTORY HDR

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>B15D</td>
<td>BUFFER IS $259</td>
</tr>
<tr>
<td>B169</td>
<td>LENGTH IS $2B (ONE ENTRY) (BED9)</td>
</tr>
<tr>
<td>B173</td>
<td>ML1: READ &lt;BE70&gt;</td>
</tr>
<tr>
<td>B176</td>
<td>ERROR? &gt;&gt; B193</td>
</tr>
<tr>
<td>B17A</td>
<td>COPY ENTRY LENGTH, ENTRIES PER BLOCK, (B27C)</td>
</tr>
<tr>
<td>B17D</td>
<td>AND FILE COUNT FROM DIR HDR (BCC7)</td>
</tr>
<tr>
<td>B183</td>
<td>STORE ENTRY LENGTH IN READ LENGTH NOW (BED9)</td>
</tr>
<tr>
<td>B188</td>
<td>SET COUNTER TO FIRST ENTRY IN BLOCK (BBCB)</td>
</tr>
<tr>
<td>B18D</td>
<td>MARK = 0 (START OF FILE) (BEC9)</td>
</tr>
<tr>
<td>B193</td>
<td>RETURN</td>
</tr>
</tbody>
</table>
BASIC Interpreter (B1) -- V1.1.1 -- 18 JUN 84  NEXT OBJECT ADDR: B193

B194 ********** OPEN XENTS
       A REGISTER = A
       X REGISTER = X

B194 ---
B198 T KEYWORD GIVEN
B199 A ACCESS BITS
B19A NO >>B19F
     DEFAULT TYPE
B19C YES, USE KEYWORD
B19F ---
B1A0 EXISTING FILE?
B1A3 NO, ERROR >>B199
B1A5 CHECK ACCESS REF VALUE INSTEAD (B86A)
B1A8 REQUESTED ACCESS
B1AA SET SYSTEM-BUFFER FOR THIS TYPE? (BEB8)
B1B2 LEVEL = $60 ($0A)
B1B7 ML: OPEN <$BE70> JESTED (B867)
B1BA ERROR? >>B1CD
     NOT PERMITTED >>B1CD
B1BF SAVE REFNUM IN EE IN OPEN PARM LIST (B868)
B1C2 AND CLOSE PARM?
B1C5 AND GET/SET EOF?
B1C8 AND EXIT
     READ/WRITE PARMLIST (B86D)
B1C9 "FILE TYPE MISMATCH (B86E)
B1CC RETURN "MARK LIST (B867)
B1CD "FILE LOCKED"
B1D0 RETURN "TEXT"

B1D1 ********** READ NF

B1D1 FORCE MARK TO ST
B1D9 CHECK ENTRY NUM
B1DE LAST ENTRY IN NEXT DIRECTORY ENTRY ***************
B1E1 NO >>B1ED
B1E4 YES, ENTRY IS NEXT OF THIS BLOCK (B869)
B1E7 BUMP MARK TO NEVER (B86B)
B1ED ---
B1E8 MARK POSITIONED?
B1F1 NO, BUMP POINTER AT TIME (B86B)
B1F4 AND CONTINUE IF T BLOCK (B869)
B1F6 JUST ENTERED SF?
B1F8 ADD 4 TO PTR TO PROPER ENTRY YET? >>B1F8
B1FF ML: SET MARK TO NEXT ENTRY (B867)
B202 ERROR? >>B21D
     STILL FIRST PAGE >>B1ED
B206 ML: READ <$BE70> ON PAGE >>B1ED
B209 ERROR? >>B21D
     ADJUST FOR BLOCK PREFIX
B20B BUMP ENTRY (COUNTER70)
B211 THIS ENTRY? W
B213 NO, SKIP OVER IT
B215 DECREMENT FILE
B21D AND RETURN TO CAGER (B860)

1CD? >>B1D1
     COUNT (B869)
     LLER
******** 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 OPEN NUM ALSO (BC9B)
B297 MLII: 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 MLII: OPEN (EXEC FILE) <BE70>

B2A7 ---
B2B1 IF ERROR, FREE BUFFER FIRST <A24C>
B2B6 THEN EXIT WITH ERROR

B2B7 SAVE BUFFNO FOR EXEC (BECF)
B2BD AND OPENNUM TOO (BED6)

******** COMPLETE EXEC COMMAND ********

B2C3 SAVE READ OPENNUM (BED6)
B2C6 AND GET/SET OPENNUM (BEC7)
B2C9 AND NEWLINE OPENNUM (BED2)
B2CF SET "L" VALUE FROM AUXID (BE5F)
B2DB SAVE PATHNAME/AUXID IN OPEN FILE TABLE <B38B>
B2DD IGNORE MSB FOR END OF LINE CHARs (BED3)
B2E2 MLII: SET NEWLINE <BE70>
B2EB WAS "P" OR "R" GIVEN ON COMMAND LINE?
B2EA NO >>B2F4
B2EC YES, POSITION TO SPECIFIED STARTING PT <B522>
B2EF NO ERRORS? >>B2F4
B2F1 IF ERROR, GO CLOSE EXEC >>B245
B2F4 MARK EXEC ACTIVE
B2FA AND RETURN TO CALLER

B2FB ********** CLOSE EXEC FILE *********

B2FB EXEC ACTIVE? (BE43) ---
B2FE NO, SKIP IT >>B30B
B30B INDICATE EXEC FILE CLOSED
B305 PICK UP REFPNUM FOR EXEC
B309 AND GO CLOSE IT <B4A5> OPENING (BE4E)
B30B RETURN FILE <BE70> (BC9B)

B30C *********** "VERIFY" COMMAND ************

B30C FILE NOT FOUND >>B347
B311 FILE FOUND, WAS A PATH?.....
B313 YES >>B31D
B315 NO, CODE61 GIVEN?
B317 PRINT "(C) APPLE COMPUTER,..." <9F8C>
B31A AND A NEW LINE <9FAB>
B31D THEN EXIT "VER...
B31E RETURN

B31F *********** FLUSH ALL OPEN ************

B31F REFPNUM = 0 (ALL OPEN): FILES
B321 JUMP INTO FLUSH >>B32F

B323 *********** "FLUSH" COMMAND **********

B323 ---
B326 WAS PATHNAME GIVEN?........
B328 NO, FLUSH ALL FILES >>B381
B32A ELSE, LOOK UP NAME IN...
B32D NOT AN OPEN FILE >>B33E<<
B32F SAVE REFPNUM IN PARM LIST OPEN FILE LISTS <B41F>
B333 MLII: FLUSH <BE70>..........
B337 EXIT...

B338 ********** "OPEN" COMMAND ********

B338 ---
B339 LOOK UP NAME IN OPEN func
B33C NOT CURRENTLY OPEN? >>B33E
B33E ---
B33F IT IS OPEN, "FILE BUSY"...
B342 RETURN ERROR
### BASIC Interpreter (BI) -- V1.1.1 -- 18 JUN 84  NEXT OBJECT ADDR: B342

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<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 (B660)</td>
</tr>
<tr>
<td>B360</td>
<td>&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;B36E</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 (BEA6)</td>
</tr>
<tr>
<td>B380</td>
<td>AND SET FILE INFO LISTS (BEBA)</td>
</tr>
<tr>
<td>B389</td>
<td>GO CREATE THE FILE &lt;AD46&gt;</td>
</tr>
<tr>
<td>B38C</td>
<td>ERROR? &gt;&gt;B349</td>
</tr>
<tr>
<td>B38E</td>
<td>CHECK FILE TYPE (BE88)</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 (BEBA)</td>
</tr>
<tr>
<td>B3A3</td>
<td>WAS &quot;L&quot; KEYWORD VALUE GIVEN?</td>
</tr>
<tr>
<td>B3A5</td>
<td>YES, USE THAT INSTEAD &gt;&gt;B3AD</td>
</tr>
<tr>
<td>B3A7</td>
<td>OTHERWISE, SAVE AUXID RECORD LEN (B660)</td>
</tr>
<tr>
<td>B3AD</td>
<td>ALLOCATE A NEW FILE BUFFER &lt;A1F5&gt;</td>
</tr>
<tr>
<td>B3B0</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>MLI: 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;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 (BE88)</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>

### BASIC Interpreter (BI) -- V1.1.1 -- 18 JUN 84  NEXT OBJECT ADDR: B3D6

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<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 REFNUM (BC9C)</td>
</tr>
<tr>
<td>B3EB</td>
<td>AND BUMP OPEN FILE COUNT AND FALL THRU (BE4D)</td>
</tr>
<tr>
<td>B3EB</td>
<td>********** SAVE FILE NAME/RECLLEN IN TABLE ***************</td>
</tr>
<tr>
<td>B3E8</td>
<td>MAKE INDEX FROM REFNUM*32 BYTES</td>
</tr>
<tr>
<td>B3F1</td>
<td>GET NAME LENGTH (B280)</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 30 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>B40B</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 (B288)</td>
</tr>
<tr>
<td>B41B</td>
<td>COPY ALL OF NAME, THEN FALL THRU TO EXIT &gt;&gt;B411</td>
</tr>
<tr>
<td>B41D</td>
<td>********** &quot;MON&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 REFNUM 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 (B64E)</td>
</tr>
<tr>
<td>B432</td>
<td>STORE FILE COUNT AS LOOP COUNTER</td>
</tr>
<tr>
<td>B434</td>
<td>GET NEXT REFNUM (BC9B)</td>
</tr>
<tr>
<td>B437</td>
<td>COMPARE FILENAMES &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 REFNUM (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 (B1) -- V1.1.1 -- 18 JUN 84 NEXT OBJECT ADDR: B446

ADDRESS DESCRIPTION/CONTENTS

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 RENUM >>B43E
B45E "FILE NOT OPEN" ERROR
B461 RETURN WITH ERROR CODE

B462 ************ COMPARISON FILENAMES ***********************
B462 RENUM*32 FOR FILENAME INDEX
B468 PICK UP DIR FLAG FROM THIS ENTRY (BCFE)
B470 SAME LENGTH AS HIS FILENAME? (02B0)
B473 NO, CAN'T BE IT THEN >>B49B
B476 MAKE SURE LENGTH DOES NOT EXCEED 29
B47A IF IT DOES, ONLY LOOK AT FIRST 29
B47C USE $A AS LOOP COUNTER
B481 COPY "$L" OF THIS FILE TO KEYWORD (BCAA)
B48A ---
B48B COMPARE NAMES (02B8)
B491 NO MATCH? EXIT WITH Z FLAG CLEAR >>B498
B498 MATCH, EXIT WITH Z FLAG SET

B499 ************ "CLOSE" COMMAND **********************
B499 ---
B49C PATHTAG? GIVEN?
B49E NO, CLOSE ALL FILES >>B4F2
B4A0 YES, LOOK IT UP IN OPEN FILE TABLES <B4F1P>
B4A3 NOT FOUND? >>B441
B4A5 FOUND IT, STORE RENUM IN CLOSE LIST (BEDE)
B4AB MARK BUFFER PAGE FREE (BC80)
B4AC EXEC CLOSING? (BE4E)
B4AF ---
B4B1 YES...NO NEED TO COMPRESS LISTS >>B4CF
B4B3 GET OPEN COUNT (LAST OPENED FILE NO.) (BE4D)
B4B7 SWAP BUFFERS (BC93)
B4C5 AND RENUMS WITH THE LAST OPENED FILE (BC9B)
B4C9 ---
B4D1 LEVEL = 0 (BF94)
B4D6 MLI: CLOSE <BE70>
B4D9 ERROR? >>B502
B4DB RELEASE THE BUFFER <A24C>
B4DE EXEC FILE CLOSING? (BE4E)
B4EC NO >>B45E
B4E6 YES, EXEC NO LONGER ACTIVE (BE43)
B4E9 AND NO LONGER CLOSING (BE4E)
B4ED RETURN TO CALLER

BASIC Interpreter (B1) -- V1.1.1 -- 18 JUN 84 NEXT OBJECT ADDR: B4ED

ADDRESS DESCRIPTION/CONTENTS

B4EE DROP OPEN FILE COUNT (BE4D)
B4F1 AND EXIT

B4F2 **************** CLOSE ALL OPEN FILES **********************
B4F2 ANY FILES OPEN? (BE4D)
B4F5 NO >>B503
B4F7 YES, EXEC NOT CLOSING (BE4E)
B4FD CLOSE LAST FILE OPENED <B4A5>
B500 IF THAT WORKS, START ALLOver AGAIN >>B4F2
B502 EXIT WHEN ALL ARE CLOSED
B503 ---
B505 SET CLOSE RENUM 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>
B513 NOT OPEN? >>B57F
B517 SET RENUM IN READ/WRITE PARMLIST (BED6)
B51A AND SET NEWLINE LIST (BED2)
B51D DIR FILE? (BE47)
B520 YES, GET OUT RIGHT NOW1 >>B59B
B522 "F" OR "R" GIVEN? (BE57)
B527 NO, INVALID PARM >>B57D
B529 BOTH GIVEN?
B52B YES, INVALID PARM >>B57D
B52D JUST "R" GIVEN?
B52F NO, JUST "F" >>B53D
B531 JUST "R", COPY "R" VALUE TO "F" (BE65)
B534 ("R" AND "F" ARE ALIASED) (BE63)
B53D SET COUNT TO 239. (MAXIMUM LINE LEN)
B54C BUFFER IS AT $200 (BE08)
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 "F" = $7? (BE64)
B562 YES, DONE >>B500
B564 ELSE...
B566 MLI: READ NEXT FIELD (LINE) <BE70>
B569 ERROR? >>B57F
B56E DECREMENT "F" VALUE BY ONE
B570 "INVALID PARAMETER" ERROR
B57F ------4 NEXT OBJECT ADDR: B57B
B580 EXIT TO CALLER

B581 ************ COMPUTE NEW FILE POSITION
(BCOMPUTES ABSOLUTE FILE POSITION -)
B585 ACCUM = CURRENT RECORD LENGTH (.9CA4)
B594 MARK = 0 (BEC9)

********** MARK = "R" * RECLEN *******

B59E SHIFT "R" VALUE RIGHT (B666) *******9**************
B5AE IF LOW BIT OFF, NO ADD >>B5BF
B5AF MARK
B5B9 ADD ONE INSTANCE OF RECLEN TO MAX
B5BB OVERFLOW? >>B5D2
B5BD ACCUM OVERFLOW? >>B5D2
B5BF SCALE ACCUM (MULTIPLIER) UP BY 2 (C)
B5CB IF "R" NOW ZERO... (B653) *******
B5CE CONTINUE LOOPING >>B59E
B5D1 ELSE, EXIT TO CALLER

B5D2 "RANGE ERROR" (BCAF)
B5D5 RETURN

B5D6 ************ "READ" COMMAND ************(AF)
B5D6 LOOK UP FILE NAME (B41F)
B5D9 NOT OPEN? >>B62B
B5DB ITS OPEN, STORE REFNUM IN READ/WRITE
B5DC GET/SET... (BEC7)
B5EE AND SET NEWLINE PARMLISTS (BED2)
B5E4 DIR FILE? (BE47)
B5E7 YES, SPECIAL HANDLING REQUIRED >>B530************(**************
B5E9 NO, PRE-POSITION FOR "B"?, "F"?, OR "S"
B5EC ERROR POSITIONING? >>B628
B5EE ASSUME "L" = 239.
B5F5 "L" GIVEN?
B5F7 NO >>B668
B5F9 YES, USE HIS "L" VALUE (B66F)
B5FF UNLESS ITS < 256 >>B661
B603 OR > 239. >>B661
B667 DOUBLE QUOTE IT SO CONNAS COME THROUGH <B666>
B6A0 READ INTO $201
B6C0 IF NO "L", READ TO $200 (BED7)
B612 NL CHAR = $6D/$6B (OR NONE IF "")
B621 NL: SET NEWLINE <BE7B>
B624 ERROR? >>B62B
B626 ---

(0200)
(BED3)

BASIC Interpreter (B1) -- V1.1.1 -- 18 JUN 84 NEXT OBJECT ADDR: B628
ADDR DESCRIPTION/CONTENTS

B62B MARK INPUT "READ" FILE ACTIVE (BE44)
B62B AND RETURN

********** READ DIR FILE **********

B62C SET READ/WRITE LIST REFNUM (BED6)
B62F AND GET/SET LIST REFNUM (BEC7)
B634 READING TO $259 (BED7)
B63B INIT LTD FLAG TO FIRST LINE VALUE (BE4F)
B644 "R" GIVEN?
B647 NO, DONE >>B626
B64B YES, ZERO OUT MARK (BEC8)
B656 NL: REWIND FILE <BE70>
B659 ERROR? >>B660
B65D MARK INPUT FILE ACTIVE (BE44)
B660 AND EXIT

B661 ************ "RANGE ERROR" ************

B661 "RANGE ERROR" CODE
B665 EXIT TO CALLER

B666 ********** PRE-POSITION FOR I/O **********

B666 ---
B669 "B", "F", OR "R" GIVEN?
B66B NO, EXIT >>B6AF
B66D "R"
B66F NO >>B67B
B671 YES, COMPUTE ABSOLUTE POSITION (B581)
B674 ERROR? >>B661
B676 NO, SET MARK TO NEW POSITION <B6A8>
B679 ERROR? >>B680
B67B "R" GIVEN? (B667)
B680 NO >>B667
B682 SKIP LINES UNTIL "F" = 0 <B63D>
B685 ERROR >>B6B8
B687 "B" GIVEN? (B667)
B68C NO >>B6AF
B690 NL: GET MARK <BE78>
B693 ERROR >>B680
B699 ADD "B" VALUE TO CURRENT MARK (BE5A)
B69C (3 BYTE ADD) (BEC8)
B6A6 OVERFLOW? >>B661
B6A8 ---
B6AA NL: SET MARK <BE78>
B6AD ERROR >>B6B0
B6AF ---
B6B0 ---
B6B2 EXIT TO CALLER
B6C0 **WRITE** COMMAND

B6C0

B6C2 LOOKUP OPEN FILE NAME <B416>
B6C4 NOT AN OPEN FILE? >>B6C8
B6C6 STORE READ/WRITE REFNUM (BED6)
B6C8 AND GET/SET REFNUM (BEC7)
B6C6 AND NEWLINE REFNUN IN PARM LISTS (BED2)
B6C6 DIR FILE? (BE47)
B6C4 NO, OK >>B6CA
B6C6 YES, "FILE LOCKED" ERROR
B6C8 --
B6CA EXIT TO CALLER
B6C0 DATA BUFFER AT $200
B6C4 PRE-POSITION FOR "B", "F", AND "R" <B666>
B6C7 NO ERRORS? >>B6ED
B6C9 WAS ERROR A RANGE ERROR?
B6D6 NO, REAL ERROR >>B6C8
B6C6 YES, MY RANGE ERROR OR MLI'S?
B6E6 MIKE... >>B6CB
B6E1 MLI'S... SET EOF FARTHER INTO FILE
B6F3 MLI: SET EOF <BE70>
B6E6 ERROR? >>B6CB
B6F3 AND THEN TRY AGAIN TO SET MARK <B676>
B6F8 ERROR? THEN I GIVE UP >>B6CB
B6F1 BUFFER IS AT HIMEN
B6F99 INDICATE OUTPUT "WRITE" FILE ACTIVE (BE45)
B6F9 RETURN TO CALLER
B6 **APPEND** COMMAND

B6C0

B6C2 LOOK UP NAME IN OPEN FILE LIST <B416>
B6C4 FOUND IT? >>B710
B6C6 NO, OPEN IT FIRST <B338>
B6C8 ERROR? >>B71E
B6E6 NO, REFNUN NON-ZERO? (BED8)
B6E0 YES, OK >>B711
B6E9 ELSE, BREAK11
B6C6 ---
B711 REFNUN TO READ/WRITE PARM LIST (BED6)
B714 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 (BE69)
B733 ---
B738 COMPUTE REFNUN*32 FOR INDEX INTO
B739 FILE NAME TABLE
B73E SAVE CURRENT "L" VALUE IN OPEN FILE (BCCF)
B741 NAME TABLE AND IN CURRENT RECLEN (BCA4)
B74D MLI: GET EOF <BE70>
B750 ERROR? >>B71E
B752 IS "L" VALUE < 27 (NO SPECIFIC "L") (BCA5)
B755 NO >>B75E
B75C YES >>B763
B75E NO, FORCE TO RECORD BOUNDARY <B766>
B761 ERROR? >>B71E
B763 ELSE, GO SET EOF=MARK/OUTPUT FILE ACTIVE >>B6E1

B766 **FORCE** TO EVEN RECORD BOUNDARY

B766

B766 FIND RECORD NUMBER OF THIS POSITION

B766

B766 COPY EOF TO ACCUM (BCC7)
B771 CLEAR MSBS (BCE2)
B777 GET READY 1 A 24 BIT DIVIDE
B779 DIVIDE EOF BY... <AAD7>
B786 RECORD LENGTH (BCA4)
B798 ---
B7A1 WAS THERE A REMAINDER? (BCB3)
B7A5 NO, OK... >>B7CF
B7AB YES, CURRENT RECORD LEN LESS REMAINDER (BCB2)
B7B0 PLUS OLD EOF MARK (BCE8)
B7C2 GIVES NEW EOF ON AN EVEN RECORD BOUNDARY (BEC9)
B7CD "RANGE ERROR" POSSIBLE IF OVERFLOW OCCURS
B7C8 RETURN TO CALLER

B7D8 **GET FILE INFO**

B7D8

B7DF SET NUMBER OF PARRS (10)
B7D5 MLI CODE FOR GET FILE INFO
B7D7 GO DO IT >>B7E6
B7D9 ******* SET FILE INFO ************

B7D9 MODIFIED TIME/DATE = 0
B7E7 SET NUMBER OF PARGS (7)
B7EC MLI CODE FOR SET FILE INFO
B7EE EXIT THRU MLI: GET/SET FILE INFO >>BE70

B7F1 ********** BI I/O INDIRECTION VECTORS **********

B7F1 DOSSOUT VECTOR >>BE38
B7F4 DOSSIN VECTOR >>BE3A

B7F7 ********** STATE I/O VECTORS TABLE **********

B7F7 IMMEDIATE MODE (STATE=0) CSWL/KSWL
B7F8 DEFERRED MODE (STATE=4) CSWL/KSWL
B7FF (STATE=8) CSWL/KSWL
B803 (STATE=C) CSWL

B805 ********** SYSCBL **********

LSB'S OF MLI CALL PARAMETER LISTS IN THE
BI GLOBAL PAGE ($BEXX)

B805 CREATE: $A0 DESTROY: $AC RENAME: $AF
B808 S.PI: $B4 G.PI: $B4 ONLINE: $C6
B80B S.PFX: $AC G.PFX: $AC OPEN: $CB
B80E NEWLINE: $D1 READ: $D5 WRITE: $DF
B811 CLOSE: $DD FLUSH: $DD SMARK: $C6
B814 GMARK: $C6 SEOF: $C6 GEOF: $C6
B817 SBUF: $C6 GBUF: $C6

B819 ********** APPLSOFT 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 $B6 (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 FRE 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 1C VERIFY 1D CATALOG 1E RESTORE
B877 1F POSITION

B878 'BESERIFSYLOADELETETBYRCATALOGOPE'
B87B 'NWRITEEXECREATEFREESTORENAMEBRUNLO'
B886 'CKCHAIN#FLUSHREADPOSITIONONMONP#'
B888 'PREFIXCLOSEAPPEND'

B889 ********** COMMAND HANDLER ADDRESS TABLE **********

ADDRESSES OF THE COMMAND HANDLER ROUTINES
FOR EACH COMMAND IN THE ORDER GIVEN ABOVE.

B889 (EXTERNAL)
B889 IN#
B88D PR#
B88F CKT
B891 FRE
B893 BYE
B895 RUN
B897 BRUN
B89F EXEC
B8A0 LOAD
B8A2 LOCK
B8A4 OPEN
B8A7 READ
B8A9 SAVE
B8A0 BLOAD
B8A7 BSAVE
B8A9 CHAIN
B8B0 CLOSE
B8B2 FLUSH
B8B4 NOMON
B8B7 STORE
B8B9 WRITE
B8BC APPEND
B8BE CREATE
B8BF DELETE
B8C1 PREFIX
B8C3 RENAME

Beneath Apple ProDOS Supplement
## BASIC Interpreter (B1) — V1.1.1 — 18 JUN 84 NEXT OBJECT ADDR: B91F

<table>
<thead>
<tr>
<th>ADDR</th>
<th>Description/Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>B91F</td>
<td>UNLOCK</td>
</tr>
<tr>
<td>B921</td>
<td>VERIFY</td>
</tr>
<tr>
<td>B923</td>
<td>CATALOG</td>
</tr>
<tr>
<td>B925</td>
<td>RESTORE</td>
</tr>
<tr>
<td>B927</td>
<td>POSITION</td>
</tr>
<tr>
<td>B929</td>
<td>&quot;-&quot; COMMAND</td>
</tr>
</tbody>
</table>

**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
- 0001 = SLOT (FOR PR# OR IN#)
- 0002 = DEFERRED COMMAND ONLY
- 1000 = FILENAME IS OPTIONAL
- 0003 = IF FILE NOT FOUND, CREATE IT
- 0400 = "S" (FILE TYPE) PERMITTED
- 0200 = PATHNAME1 (RENAME) PERMITTED
- 0100 = PATHNAME1 EXPECTED
- 0004 = "A" (ADDRESS) PERMITTED
- 0005 = "B" (BYTE) PERMITTED
- 0010 = "E" (END ADDRESS) PERMITTED
- 0011 = "L" (LENGTH) PERMITTED
- 0012 = "E" (LINE No.) PERMITTED
- 0007 = "A" (ADDRESS) PERMITTED
- 0008 = "S" (SLOT) PERMITTED
- 0009 = "O" (OFFSET) PERMITTED
- 0010 = "R" (RECORD) PERMITTED
- 0100 = "F" (FIELD) PERMITTED
- 0110 = "P" (PARAMETER) PERMITTED
- 0200 = "E" (END ADDRESS) PERMITTED

("V" IS IGNORED)

<table>
<thead>
<tr>
<th>C</th>
<th>P</th>
<th>S</th>
<th>D</th>
<th>F</th>
<th>N</th>
<th>T</th>
<th>P</th>
<th>P</th>
<th>A</th>
<th>B</th>
<th>E</th>
<th>L</th>
<th>S</th>
<th>F</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>O</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>B92B</th>
<th>IN#</th>
</tr>
</thead>
<tbody>
<tr>
<td>B92D</td>
<td>PR#</td>
</tr>
<tr>
<td>B92F</td>
<td>CAT</td>
</tr>
<tr>
<td>B931</td>
<td>FRE</td>
</tr>
<tr>
<td>B933</td>
<td>BYE</td>
</tr>
<tr>
<td>B935</td>
<td>RUN</td>
</tr>
<tr>
<td>B937</td>
<td>BRUN</td>
</tr>
<tr>
<td>B939</td>
<td>EXEC</td>
</tr>
<tr>
<td>B93B</td>
<td>LOAD</td>
</tr>
<tr>
<td>B93D</td>
<td>LOCK</td>
</tr>
<tr>
<td>B93F</td>
<td>OPEN</td>
</tr>
<tr>
<td>B941</td>
<td>READ</td>
</tr>
<tr>
<td>B943</td>
<td>SAVE</td>
</tr>
<tr>
<td>B945</td>
<td>LOAD</td>
</tr>
<tr>
<td>B947</td>
<td>BSAVE</td>
</tr>
</tbody>
</table>

**B949 CHInterpreter (B1) — V1.1.1 — 18 JUN 84 NEXT OBJECT ADDR: B949**

<table>
<thead>
<tr>
<th>ADDR</th>
<th>Description/Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>B949</td>
<td>CHInterpreter</td>
</tr>
<tr>
<td>B94B</td>
<td>CLOCK</td>
</tr>
<tr>
<td>B94D</td>
<td>FFR/DESCRIPTION/CONTENTS</td>
</tr>
<tr>
<td>B94F</td>
<td>NOM</td>
</tr>
<tr>
<td>B951</td>
<td>STC</td>
</tr>
<tr>
<td>B953</td>
<td>WRITEN</td>
</tr>
<tr>
<td>B955</td>
<td>AP13E</td>
</tr>
<tr>
<td>B957</td>
<td>CB1SH</td>
</tr>
<tr>
<td>B959</td>
<td>DEF1W</td>
</tr>
<tr>
<td>B95B</td>
<td>PR1RE</td>
</tr>
<tr>
<td>B95D</td>
<td>REM1G</td>
</tr>
<tr>
<td>B95F</td>
<td>UNEND</td>
</tr>
<tr>
<td>B961</td>
<td>VE1ATE</td>
</tr>
<tr>
<td>B963</td>
<td>CA1FTE</td>
</tr>
<tr>
<td>B965</td>
<td>RES1F</td>
</tr>
<tr>
<td>B967</td>
<td>BW1X vm</td>
</tr>
<tr>
<td>B969</td>
<td>&quot;-&quot; CM</td>
</tr>
<tr>
<td>B96B</td>
<td>&quot;-&quot; CM</td>
</tr>
<tr>
<td>B968</td>
<td>STORE</td>
</tr>
</tbody>
</table>

**B975***** KEYWORD NAME TABLE ******************************

| NAME | CDLSDFPRGv| |
|------|------------|

**B975***** KEYWORD BIT POSITION TABLE ******************************

**B97F***** FOR EACH KEYWORD IN THE ORDER GIVEN IN NAME TABLE. "V" IS 00 (NOT USED)**

**B97F FOR KEYWORD SIZE/OFFSET TABLE ******************************

<table>
<thead>
<tr>
<th>B988</th>
<th>B: 2 BITS — SIZE-1 OF VALUE IN BYTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>B991</td>
<td>E: High 6 BITS—OFFSET TO LAST BYTE OF VALUE FROM $BE58</td>
</tr>
</tbody>
</table>

**B992 L1 |

**B993 S: |

**B994 R: 2 BYTES AT +1 |

**B995 F: 3 BYTES AT +4 |

**B996 R: 2 BYTES AT +6 |

**B997 V: 2 BYTES AT +9 |

**B998 $1: 2 BYTES AT +9 |

**B999 R: 2 BYTES AT +C |

**B999 R: 2 BYTES AT +E |

**B999 R: 2 BYTES AT +10 (IGNORED) |

**B999 R: 2 BYTES AT +11 |

**FILE TYPES TABLES ******************************

FILE TYPE CODES, GIVEN IN INVERSE ORDER TO FILE TYPE NAMES WHICH FOLLOW.
BASIC Interpreter (BI) -- V1.1.1 -- 18 JUN 84  NEXT OBJECT ADDR: B989

--- DESCRIPTION/CONTENTS ---

B989 $FL - OPTION/CONTENTS
B98A $F2 -
B98B $F3 -
B98C $F4 -
B98D $F5 - "SYS"
B98E $F6 - "HEL"
B98F $F7 - "VAN"
B990 $F8 - "BAS"
B991 $F9 - "IVR"
B992 $FA - "INT"
B993 $FB - "GMD"
B994 $FC - "DIR"
B995 $FD - "BIN"
B996 $FE - "TXT"
B997 $FF - "ASP"
B997 $70 - "AWP"
B997 $71 - "ABB"

---

BA38  ********** LESS COMMON LETTERS ************
BA39  'GHKPSVWXY()_.'

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($4B)
BA74  "ENDFILE SUBTYPE"
BA7E  "BLOCKS FREE;"
BA86  TAB($16)
BA88  "BLOCKS USED;"
BA91  TAB($2C)
BA93  "TOTAL BLOCKS;"

BA9C  "RANGE ERROR"  ERROR=$12
BA93  "NO DEVICE CONNECTED"  ERROR=$13
BAAE  "WRITE PROTECTED"  ERROR=$14
BAB7  "END OF DATA"  ERROR=$15
BABD  "PATH NOT FOUND"  ERROR=$16,
BAC9  "I/O ERROR"  ERROR=$18
BACC  "DISK FULL"  ERROR=$19
BAD2  "FILE LOCKED"  ERROR=$1A
BAD9  "INVALID PARAMETER"  ERROR=$1B
BAE3  "RAM TOO LARGE"  ERROR=$1C
BAF9  "FILE TYPE MISMATCH"  ERROR=$1D
### BASIC Interpreter (BI) -- V1.1.1 -- 16 JUN 84 NEXT OBJECT ADDR: BAF8

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAF0</td>
<td>&quot;PROGRAM TOO LARGE&quot;  ERROR=$E</td>
</tr>
<tr>
<td>BB07</td>
<td>&quot;NOT DIRECT COMMAND&quot;  ERROR=$F</td>
</tr>
<tr>
<td>BB11</td>
<td>&quot;SYNTAX ERROR&quot;        ERROR=$10</td>
</tr>
<tr>
<td>BB19</td>
<td>&quot;DIRECTORY FULL&quot;      ERROR=$11</td>
</tr>
<tr>
<td>BB21</td>
<td>&quot;FILE NOT OPEN&quot;        ERROR=$12</td>
</tr>
<tr>
<td>BB29</td>
<td>&quot;DUPLICATE FILE NAME&quot;  ERROR=$13</td>
</tr>
<tr>
<td>BB34</td>
<td>&quot;FILE BUSY&quot;            ERROR=$14</td>
</tr>
<tr>
<td>BB3B</td>
<td>&quot;FILE(S) STILL OPEN&quot;   ERROR=$15</td>
</tr>
<tr>
<td>BB47</td>
<td>********** VARIABLES  **********************************************</td>
</tr>
<tr>
<td>BB47</td>
<td>NUMBER OF PAGES TO ALLOCATE/FREE</td>
</tr>
<tr>
<td>BB48</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>
<tr>
<td>BB4B</td>
<td>********** $BB4B-$BC7A NOT USED  **********************************************</td>
</tr>
<tr>
<td>BB4B</td>
<td>NOT USED</td>
</tr>
<tr>
<td>BC7B</td>
<td>********** VARIABLES  **********************************************</td>
</tr>
<tr>
<td>BC7B</td>
<td>SAVED HIMEM VALUE DURING CHAIN LOAD</td>
</tr>
<tr>
<td>BC7C</td>
<td>GC: HIRANGE = WORKAREA SIZE</td>
</tr>
<tr>
<td>BC7D</td>
<td>GC: WORKAREA MSB</td>
</tr>
<tr>
<td>BC7E</td>
<td>GC: NUMBER OF PAGES IN WORKAREA</td>
</tr>
<tr>
<td>BC7F</td>
<td>GC: LOHANGE (START OF STRINGS TO COPY)</td>
</tr>
<tr>
<td>BC80</td>
<td>GC: HIRANGE (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>
<tr>
<td>BC89</td>
<td>********** STORED VARIABLES FILE HEADER ***</td>
</tr>
<tr>
<td>BC8B</td>
<td>COMBINED LEN OF SIMPLE/ARRAY VARS</td>
</tr>
<tr>
<td>BC8B</td>
<td>LEN OF SIMPLE VARS ONLY</td>
</tr>
<tr>
<td>BC8D</td>
<td>HIMEM WHEN VARS WERE COMBINED  **********************************************</td>
</tr>
<tr>
<td>BC8B</td>
<td>POINT TO COMBINED VARIABLES/STRINGS</td>
</tr>
<tr>
<td>BC90</td>
<td>LENGTH OF COMBINED VARIABLES/STRINGS</td>
</tr>
<tr>
<td>BC92</td>
<td>LENGTH OF STRINGS ONLY</td>
</tr>
</tbody>
</table>

### BASIC Interpreter (BI) -- V1.1.1 -- 16 JUN 84 NEXT OBJECT ADDR: BC92

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>BC94</td>
<td>OPEN FILES' BUFFER MSB</td>
</tr>
<tr>
<td>BC9B</td>
<td>OPEN EXEC FILE BUFFER MS</td>
</tr>
<tr>
<td>BC9C</td>
<td>OPEN FILES' REFERENCE MSB</td>
</tr>
<tr>
<td>BC9D</td>
<td>OPEN EXEC FILE REFNUM</td>
</tr>
<tr>
<td>BCA4</td>
<td>CURRENT RECORD LENGTH</td>
</tr>
<tr>
<td>BCA6</td>
<td>NOT USED</td>
</tr>
<tr>
<td>BCA9</td>
<td>CHARACTER TO FLUSH WHEN</td>
</tr>
<tr>
<td>BCAA</td>
<td>MAXIMUM LENGTH TO PARSE</td>
</tr>
<tr>
<td>BCA8</td>
<td>ADDRESS OF COMMAND HANDLING ROUTINE</td>
</tr>
<tr>
<td>BCAD</td>
<td>SIZE OF KEYWORD VALUE PADDING BLANK</td>
</tr>
<tr>
<td>BCAE</td>
<td>OFFSET INTO KEYWORD PADDING BLANK ROUTINE</td>
</tr>
<tr>
<td>BCAF</td>
<td>GENERAL PURPOSE 4 BYTE NAME 5 BYTE NAME</td>
</tr>
<tr>
<td>BC83</td>
<td>MONTH</td>
</tr>
<tr>
<td>BC84</td>
<td>DAY</td>
</tr>
<tr>
<td>BC85</td>
<td>YEAR</td>
</tr>
<tr>
<td>BC8B</td>
<td>ERROR MSG LEN OR LINE</td>
</tr>
<tr>
<td>BC87</td>
<td>ENTRY LENGTH IN DIRECT INDEX</td>
</tr>
<tr>
<td>BC88</td>
<td>ENTRIES PER BLOCK IN DIRECT INDEX</td>
</tr>
<tr>
<td>BC89</td>
<td>FILE COUNT FROM DIRECT INDEX</td>
</tr>
<tr>
<td>BC98</td>
<td>DIRECTORY ENTRY NUMBER 0-1024</td>
</tr>
<tr>
<td>BC9B</td>
<td>DIRECTORY FILE</td>
</tr>
<tr>
<td>BC9C</td>
<td>PATHNAME 1 BUFFER</td>
</tr>
<tr>
<td>BCBC</td>
<td>COMMAND OR PATH LENGTH</td>
</tr>
<tr>
<td>BCBD</td>
<td>TXSBUF (COMMAND OR PATH)</td>
</tr>
<tr>
<td>BCFD</td>
<td>NOT USED</td>
</tr>
<tr>
<td>BCFE</td>
<td>********** OPEN FILE NAME (X STRING)</td>
</tr>
<tr>
<td>BCFE</td>
<td>(EACH ENTRY IS 32 BYTES)</td>
</tr>
<tr>
<td>BCFF</td>
<td>(THERE ARE 8 ENTRIES)  **********************************************</td>
</tr>
<tr>
<td>BC90</td>
<td>FILE 0: LENGTH OF NAME MSG LONG</td>
</tr>
<tr>
<td>BD00</td>
<td>FILE 0: L VALUE LSB</td>
</tr>
<tr>
<td>BD01</td>
<td>FILE 0: L VALUE MSB</td>
</tr>
<tr>
<td>BD01</td>
<td>FILE 0: START OF NAME</td>
</tr>
<tr>
<td>BD0E</td>
<td>LAST 2 BYTES NOT USED  **********************************************</td>
</tr>
</tbody>
</table>
Beneath Apple ProDOS Supplement

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".
### ProDOS BI Global Page

<table>
<thead>
<tr>
<th>ADDR</th>
<th>LABEL</th>
<th>CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>BE53</td>
<td>XCNUM</td>
<td>Number of command:</td>
</tr>
<tr>
<td>$80 = external</td>
<td>$80 = OPEN</td>
<td>$14 = WRITE</td>
</tr>
<tr>
<td>$81 = IN#</td>
<td>$80 = READ</td>
<td>$15 = APPEND</td>
</tr>
<tr>
<td>$82 = $4C = SAVE</td>
<td>$16 = CREATE</td>
<td></td>
</tr>
<tr>
<td>$83 = CAT</td>
<td>$80 = BLOAD</td>
<td>$17 = DELETE</td>
</tr>
<tr>
<td>$84 = UFRE</td>
<td>$80 = BSAVE</td>
<td>$18 = PREFIX</td>
</tr>
<tr>
<td>$85 = RUN</td>
<td>$80 = CHAIN</td>
<td>$19 = RENAME</td>
</tr>
<tr>
<td>$86 = BRUN</td>
<td>$10 = CLOSE</td>
<td>$1A = UNLOCK</td>
</tr>
<tr>
<td>$87 = EXEC</td>
<td>$11 = FLUSH</td>
<td>$1B = VERIFY</td>
</tr>
<tr>
<td>$88 = LOAD</td>
<td>$12 = NOMON</td>
<td>$1C = CATALOG</td>
</tr>
<tr>
<td>$89 = SAVE</td>
<td>$13 = STORE</td>
<td>$1D = RESTORE</td>
</tr>
<tr>
<td>$1E = POSITION</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### BE54-BE55 PBITS

Permitted command operands bits:

- $4000: Slot number only (PR# or IN#).
- $2000: Deferred command.
- $1000: File name optional.
- $8000: If file does not exist, create it.
- $4000: T: file type required.
- $2000: Second file name required.
- $1000: First file name required.
- $8000: AD: address keyword permitted.
- $4000: B: byte offset permitted.
- $2000: E: ending address permitted.
- $1000: L: length permitted.
- $8000: #: line number permitted.
- $4000: S or D: slot/drive permitted.
- $1000: R: record permitted.

(V always permitted but ignored.)

### BE56-BE57 PBITS

Operands found on command line. Same bit assignments as above.

- BE58-BE59: VADDR
- BE5A-BE5C: VBYTE
- BE5D-BE5E: VENDA
- BE5F-BE60: VLTH
- BE61: VSLT
- BE62: VDRIV
- BE63-BE64: VULED
- BE65-BE66: VRXCD
- BE67: VVOLM
- BE68-BE69: VLINE
- BE6A: VTYPE
- BE6B: VIOSLT

### BE6C-BE6D: VPATH1

Primary pathname buffer (address of length byte).

### BE6E-BE6F: VPATH2

Secondary pathname buffer (address of length byte).

### BE70-BE84: GOSYSTEM

Call the MLI using the parameter tables which follow.

### BE85

MLI call number for this call.

### BE86-BE87: SYSFARM

Address of MLI parameter list for this call.

### BE88-BE8A

Return from MLI call.

### BE8B-BE9E: BADCALL

MLI error return: translate error code to BI error number.

### BE9F

Not used.

### BEAF-BEAE: SCKSCRE

CREATE parameter list.

### BEAC-BEAE: SCKPCFX

GET_LAST, SET_FNAME, DESTROY parameter list.

### BEAF-BEBC: SRENAM

RENAME parameter list.

### BEB4-BEB5: SSINF

GET_FILE_INFO, SET_FILE_INFO parameter list.

### BEC5-BECA: SONLINE

ONLINE, SET MARK, GET_MARK, SET_EOF, GET_EOF, SET_BUF, GET_BUF, QUIT parameter list.

### BECB-BED0: SOPEM

OPEN parameter list.

### BED1-BED4: SNEWLN

SET_NEWLINE parameter list.

### BED5-BED7: SREAD

READ, WRITE parameter list.

### BEDD-BEDE: SCLOSE

CLOSE, FLUSH parameter list.

### BEFD-BEFE: CCCUSPAR

"COPYRIGHT APPLE, 1983" GETBUFPR buffer allocation subroutine vector.

### BEFF-BEFF: FREEBUF

FREEBUF buffer free subroutine vector.

### BEFB

Original HIMEM MSB.

Not used.
**Beneath Apple ProDOS Supplement**

**Disk Controller Boot ROM -- Apple II/Ill/Ile**  
**NEXT OBJECT ADDR: C600**

**ADDRESS DESCRIPTION/CONTENTS**

**C600** MODULE STARTING ADDRESS

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

$026 SECTOR BUFFER POINTER
$02B SLOT NUMBER * 16 FOR INDEX
$03C WORKBYTE
$03D SECTOR WANTED
$040 TRACK FOUND
$041 TRACK WANTED

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

$010 SYSTEM STACK
$000 AUXILIARY BUFFER
$035 TRANSLATE TABLE
$080 SECTORS TO LOAD
$081 ENTRY POINT
$088 PHASES OFF
$090 PHASES ON
$099 MOTOR ON
$0BA DRIVE SELECT
$0BE READ DATA REGISTER
$0BE SET READ MODE
$0CA8 MONITOR WAIT ROUTINE
$0F58 RTS

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

C600 SIGNATURE
C602 INITIALIZE TABLE VALUE INDICATOR
C606 STORE BIT PATTERN
C609 SHIFT PATTERN LEFT ONE BIT
C60A ARE THERE ANY TWO ADJACENT BITS ON?
C60C NO, TRY ANOTHER PATTERN >>C61E
C60E YES, TURN OFF RIGHTMOST EACH GROUP OF ZEROES
C610 FLIP BITS, PAIR OF ZERO BITS NOW SINGLE ONE BIT
C612 HIGH BIT ALWAYS ON/TURN OFF BIT WE MISSED BEFORE
C614 ---- >>C61E
C616 SHIFT PATTERN RIGHT, MUST HAVE ONLY ONE BIT ON

C617 IF MORE THAN ONE BIT ON, TRY ANOTHER PATTERN >>C614
C619 FOUND ONE, GET TABLE VALUE
C61A AND STORE IT IN TABLE ($356)
C61D INCREMENT TABLE VALUE INDICATOR
C61E GET NEXT BIT PATTERN, DONE YET
C61F NO, GO CHECK IT OUT >>C666

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

C621 CALL A KNOWN RTS <$F58>
C624 GET STACK POINTER
C628 GET HIGH BYTE OF WHERE WE ARE ($180)
C62D TIMES 16 TO GET SLOT
C62E SAVE SLOT
C62F PUT IN X REG FOR INDEX
C635 SELECT DRIVE 1 ($06A)
C638 TURN THE MOTOR ON ($069)

C63D ************ RECALIBRATE DISK ARM ************

C63D PREPAIR TO STEP THE ARM 80 PHASES
C63F TURN A PHASE OFF ($080)
C644 PUT COUNTER IN ACCUMULATOR
C641 CREATE A PHASE NUMBER (1-3)
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
C658 LOOP UNTIL ALL 80 ARE DONE >>C63D

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

C652 ----
C654 SECTOR TO FIND -> $80
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 ************

C658 CHECK DATA REGISTER ($08C)
C661 LOOP UNTIL DATA IS VALID >>C65E
C663 IS IT A $D57?
C665 NO, TRY AGAIN >>C65E
C667 YES, CHECK REGISTER AGAIN ($08C)
C66A LOOP UNTIL VALID >>C667
C66C IS IT AN $AA
Beneath Apple ProDOS Supplement

Disk Controller Boot ROM -- Apple II

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>C66E</td>
<td>NO, SEE IF ITS A 905 &gt;&gt;C66</td>
</tr>
<tr>
<td>C670</td>
<td>YES, DELAY FOR REGISTER TO</td>
</tr>
<tr>
<td>C671</td>
<td>CHECK REGISTER (C88C)</td>
</tr>
<tr>
<td>C674</td>
<td>LOOP UNTIL VALID &gt;&gt;C671</td>
</tr>
<tr>
<td>C676</td>
<td>IS IT A 996</td>
</tr>
<tr>
<td>C678</td>
<td>YES, WE FOUND AN ADDRESS HE</td>
</tr>
<tr>
<td>C67A</td>
<td>NO, WE FOUND ONE PREVI</td>
</tr>
<tr>
<td>C67B</td>
<td>IF NOT, START OVER &gt;&gt;C65C</td>
</tr>
<tr>
<td>C67D</td>
<td>WAS IT AN 9AD?</td>
</tr>
<tr>
<td>C67F</td>
<td>YES, WE FOUND A DATA HEADER,</td>
</tr>
<tr>
<td>C681</td>
<td>NO, START OVER &gt;&gt;C65C</td>
</tr>
<tr>
<td>C683</td>
<td>--------- DECODE ADDRESS FILE</td>
</tr>
<tr>
<td>C685</td>
<td>INITIALIZE COUNTER</td>
</tr>
<tr>
<td>C686</td>
<td>SAVE VALUE DECODED, WILL BE</td>
</tr>
<tr>
<td>C687</td>
<td>READ DATA REGISTER (C88C) CLEAR</td>
</tr>
<tr>
<td>C688</td>
<td>LOOP UNTIL DATA VALID &gt;&gt;C688</td>
</tr>
<tr>
<td>C689</td>
<td>SHIFT Bits INTO POSITION X</td>
</tr>
<tr>
<td>C68D</td>
<td>SAVE FOR LATER</td>
</tr>
<tr>
<td>C68F</td>
<td>READ REGISTER FOR NEXT BYT</td>
</tr>
<tr>
<td>C692</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>C69F</td>
<td>IS IT TRACK WE WANT?</td>
</tr>
<tr>
<td>C6A0</td>
<td>NO, START OVER &gt;&gt;C65C</td>
</tr>
<tr>
<td>C6A3</td>
<td>YES, ADDRESS FOUN</td>
</tr>
<tr>
<td>C6A6</td>
<td>----------- READ DATA FIELD</td>
</tr>
<tr>
<td>C6AC</td>
<td>INITIALIZE OFFSET (AUXILIARY (C08C) )</td>
</tr>
<tr>
<td>C6A8</td>
<td>------</td>
</tr>
<tr>
<td>C6AA</td>
<td>READ DATA REGISTER (C08C) X1X AND X1X1X1X</td>
</tr>
<tr>
<td>C6AD</td>
<td>LOOP UNTIL VALID &gt;&gt;C6AD</td>
</tr>
<tr>
<td>C6AF</td>
<td>EXCLUSIVE-OR WITH TRANSLAT</td>
</tr>
<tr>
<td>C6B4</td>
<td>DECREMENT OFFSET</td>
</tr>
<tr>
<td>C6B5</td>
<td>STORE BYTE IN AUXILIARY BU</td>
</tr>
<tr>
<td>C6B8</td>
<td>LOOP UNTIL BUFFER FULL &gt;&gt;</td>
</tr>
<tr>
<td>C6BA</td>
<td>INITIALIZE OFFSET (MAIN BU</td>
</tr>
<tr>
<td>C6BC</td>
<td>READ DATA REGISTER (C88C)</td>
</tr>
<tr>
<td>C6BF</td>
<td>LOOP UNTIL VALID &gt;&gt;C6BC</td>
</tr>
<tr>
<td>C6C1</td>
<td>EXCLUSIVE-OR WITH TRANSLATE</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;</td>
</tr>
<tr>
<td>C6CB</td>
<td>READ DATA REGISTER (C88C)</td>
</tr>
</tbody>
</table>

Disk Controller Boot ROM -- Apple II/II+/IIe

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>C6CE</td>
<td>LOOP UNTIL</td>
</tr>
<tr>
<td>C6D0</td>
<td>IS CHECKSUM</td>
</tr>
<tr>
<td>C6D3</td>
<td>NO, START</td>
</tr>
<tr>
<td>C6E1</td>
<td>ROLL IN 32G</td>
</tr>
<tr>
<td>C6E6</td>
<td>SAVE COMFIR</td>
</tr>
<tr>
<td>C6E8</td>
<td>INCREMENT</td>
</tr>
<tr>
<td>C6EB</td>
<td>OFFSETS</td>
</tr>
<tr>
<td>C6EE</td>
<td>AX Z</td>
</tr>
<tr>
<td>C6F1</td>
<td>IS THERE</td>
</tr>
<tr>
<td>C6F6</td>
<td>YES, GO D</td>
</tr>
<tr>
<td>C6FB</td>
<td>5 BYTES A</td>
</tr>
<tr>
<td>C6FE</td>
<td>INITIALIZE</td>
</tr>
<tr>
<td>C6FF</td>
<td>IF THERE IS MORE TO D</td>
</tr>
<tr>
<td>C6FD</td>
<td>INCREMENT</td>
</tr>
</tbody>
</table>
Disk Controller Boot ROM -- Apple IIc

---

Next Object Addr: C552

Addr Description/Contents
---

C552 Module Starting Address
---

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

* Boot ROM - Apple IIc 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 Slot Max Retry Count (High Byte)
0026 Sector Buffer Pointer
002B Slot Number * 16 For Index
003C Workbyte
003D Sector Wanted
0040 Track Found
0041 Track Wanted
004F Drive To Boot From

**********

External Addresses **********
---

0300 Auxiliary Buffer
0356 Translate Table
07DB Screen Location
0800 Sectors To Load
0801 Entry Point
0808 Phase 0 Off
0888 Motor Off
0899 Motor On
089C Read Data Register
08E8 Set Read Mode
08EA Drive Select
0CA9 Monitor Wait Routine

---

Disk Controller Boot ROM -- Apple IIc

---

Next Object Addr: C552

Addr Description/Contents
---

C552 Slot 5 Logic In Here

---

C56F Skip Over Miscellaneous Code *************

C552 Boot Fail *************

The following two routines are in the $C500
area but are used by the $C600 Logic.

C552 Boot Fail *************

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 (07DB)
C55D Then Go To Sleep >>C55D

C55F 'Check Disk Drive'

C56F Skip Over Miscellaneous Code *************

C56F Slot 5 Logic In Here

---

C58E Build Read Translate Table *************

C58E Initialize Bit Pattern
C590 Initialize Table Value Indicator
C592 Store Bit Pattern
C595 Shift Pattern Left One Bit
C596 Are There Any Two Adjacent Bits On?
C59B No, Try Another Pattern >>C55A
C59A Yes, Turn Off Right Most Of Each Group Of Zeros
C59C Flip Bits, Pair Of Zero Bits Now Single Bit, Etc
C59B High Bit Always On/Turn Off Bit We Missed Before
C5A0 ---- >>C55A
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
C5A5 And Store It In Table ($356)
C5AA Increment Table Value Indicator
C5AB Get Next Bit Pattern, Done Yet?
C5A8 No, Go Check It Out >>C592
C5AD Main Buffer Pointer ($26) >> $8000
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

**ADDRESSES DESCRIPTION/CONTENTS**

---

**C5F5** JUMP TO BOOTFAIL

**C5F8** REMAINING 8 BYTES NOT USED BY DISK II

**C600** **-------** INITIALIZATION

**C602** SET DRIVE -> 1

**C604** INITIALIZE RETRY COUNT (HIGH BYTE)

**C608** **-------** SELECT DRIVE AND TURN IT ON

**C60B** INITIALIZE SLOT (6)

**C60D** INITIALIZE DEVICE (1 OR 2)

**C60F** SAVE DRIVE NUMBER ON STACK

**C610** INSURE READ MODE (C08E)

**C616** GET DRIVE NUMBER BACK

**C617** SELECT APPROPRIATE DRIVE (C0EA)

**C61A** TURN MOTOR ON (C089)

**C61D** **-------** RECALIBRATE DISK ARM

**C61F** PREPARE TO STEP THE ARM 80 PHASES

**C621** TURN A PHASE OFF (C080)

**C623** CREATE A PHASE NUMBER (0-3)

**C625** DOUBLE IT FOR PROPER INDEX

**C626** COMBINE WITH SLOT FOR FINAL INDEX

**C628** PUT INDEX IN X REGISTER

**C629** TURN A PHASE ON (C081)

**C62C** DELAY ABOUT 20 MICROSECONDS

**C631** DECREMENT COUNTER

**C632** LOOP UNTIL ALL 80 ARE DONE

**C634** **-------** INITIALIZATION

**C636** SECTOR TO FIND -> $00

**C63B** TRACK TO FIND -> $00

**C63A** BUILD THE TRANSLATE TABLE

**C63D** **-------** COUNT RETRIES AND INDICATE ERROR IF BOOT FAILS

**C63D** INITIALIZE RETRY COUNT

**C63F** CLEAR THE CARRY

**C640** PUSH STATUS ON STACK

**C641** KEEP STACK CLEAN

**C642** GET SLOT

---

Disk Controller Boot ROM -- Apple IIC

**ADDRESSES DESCRIPTION/CONTENTS**

---

**C644** DECREMENT RETRY COUNT, TRY AGAIN?

**C646** YES, GO DO IT

**C648** NO, TURN DRIVE OFF

**C64B** AUTO BOOT FROM SLOT?

**C64F** NO, FAIL NOW

**C651** MAYBE SLOT 5 WILL TALK TO US

**C654** TWO BYTES NOT USED

**C656** ---

**C657** DECREMENT RETRY COUNT (LOW BYTE)

**C658** IF NOT ZERO, TRY AGAIN

**C65A** IF SO,GO DECREMENT RETRY COUNT (HIGH BYTE)

**C65C** SPACE FILLER TO POSITION CODE BELOW

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

**C661** CHECK DATA REGISTER

**C663** IS IT A $05?

**C665** NO, TRY AGAIN

**C667** YES, CHECK REGISTER AGAIN

**C66A** LOOP UNTIL VALID

**C66C** IS IT AN $AA?

**C66E** NO, SEE IF ITS A $05

**C670** YES, DELAY FOR REGISTER TO CLEAR

**C671** CHECK REGISTER

**C674** LOOP UNTIL VALID

**C676** IS IT A $96

**C678** YES, WE FOUND AN ADDRESS HEADER

**C67A** NO, HAVE WE FOUND ONE PREVIOUSLY?

**C67B** IF NOT, START OVER

**C67D** WAS IT AN $AD?

**C67F** YES, WE FOUND A DATA HEADER

**C681** NO, START OVER

---

**C683** **-------** DECODE ADDRESS FIELD

**C683** INITIALIZE COUNTER

**C685** SAVE VALUE DECODED, WILL BE TRACK ON LAST PASS

**C687** READ DATA REGISTER

**C68A** LOOP UNTIL DATA VALID

**C68C** SHIFT BITS INTO POSITION X1X1X1X1

**C68D** SAVE FOR LATER

**C68F** READ REGISTER FOR NEXT BYTE

**C692** LOOP UNTIL VALID

**C694** COMBINE WITH PREVIOUS X1X1X1X1 AND X1X1X1X1

**C696** DECREMENT COUNTER, DONE YET?

**C697** NO, DO ANOTHER

**C699** KEEP THE STACK CLEAN

**C69A** IS THIS SECTOR WE WANT?

**C69C** NO, START OVER

**C69E** GET TRACK FOUND
Disk Controller Boot ROM -- Apple IIc

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>C6A8</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>
</tbody>
</table>

**C6A6**
- INITIALIZE OFFSET (AUXILIARY BUFFER)
- DECIRONMENT OFFSET
- READ DATA REGISTER (C08C)
- LOOP UNTIL VALID >>C66A
- EXCLUSIVE-OR WITH TRANSLATE TABLE (02D6)
- STORE BYTE IN AUXILIARY BUFFER (0300)
- LOOP UNTIL BUFFER FULL >>C66B
- INITIALIZE OFFSET (MAIN BUFFER)
- READ DATA REGISTER (C08C)
- LOOP UNTIL VALID >>C6CB
- IS CHECKSUM OKAY? (02D6)
- NO, START OVER >>C6A2

**C6D5**
- MERGE MAIN AND AUXILIARY BUFFERS

**C6D5**
- INITIALIZE OFFSET (MAIN BUFFER)
- INITIALIZE OFFSET (AUXILIARY BUFFER)
- DECIRONMENT OFFSET (AUX BUFFER)
- IF LESS THAN ZERO RESET IT >>C6D7
- GET BYTE FROM MAIN BUFFER
- ROLL IN TWO BITS FROM AUXILIARY BUFFER
- SAVE COMPLETED DATA BYTE
- INCREMENT OFFSET (MAIN BUFFER)
- LOOP UNTIL WHOLE BUFFER IS DONE >>C6D9

**C6E8**
- DETERMINE IF THERE IS MORE TO DO

**C6E8**
- INCREMENT MAIN BUFFER POINTER
- INCREMENT SECTOR NUMBER
- IS THERE ANOTHER SECTOR TO LOAD? (0809)
- YES, GO DO IT >>C6D3
- NO, ENTER CODE WE JUST LOADED >>0801

**C6FB**
- 5 ZERO BYTES AT END OF PAGE
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 #7F ISOLATE SLOT
CMP #30 SLOT = 3?
BEQ GOTSST YES, CONTINUE
DEX
BPL DEVLP CONTINUE SEARCH BACKWARDS
BMI NORAM CAN'T FIND IT IN DEVLST
GOTSST LDA $BF32+1,X GET NEXT NUMBER
STA $BF32,X AND MOVE THEM FORWARD
INX
CPX $BF31 REACHED LAST ENTRY?
BNE GOTSST 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:

```assembly
* 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 INSOOUT YES, SKIP IT
       DEX
       BPL INSLP KEEP UP THE SEARCH
SKP 1
* RESTORE THE DEVNUM TO THE LST
SKP 1
INSTALL LDX $BF31 GET DEVCNT AGAIN
INSLP CPX #$0D DEVICE TABLE FULL?
       BNE INSLP2
ERROR ... YOUR ERROR ROUTINE
       ... INSLP2
INSLP2 LDA $BF32-1,X MOVE ALL ENTRIES DOWN
       STA $BF32,X TO MAKE ROOM AT FRONT
       DEX FOR A NEW ENTRY
       BNE INSLP2
       LDA #$B0
       STA $BF32 SLOT 3, DRIVE 2 AT TOP OF LIST
       INC $BF31 UPDATE DEVCNT
       SKP 1
```

Page 7–26:

Modifying the ProDOS Disk II Device Driver to allow 320 blocks instead of the normal 280. The fourth command line should read:

```
520D:40
```

Modifying FILER to format 40 tracks instead of 35. The fourth command line should read:

```
4244:40
```

[See Second printing errata for information about versions other than 1.0.1]
Page 8-6:

Under "device Information", make the following changes:

BF10-BF11  DEVAADR01  Slot 0 reserved.
...
BF26-BF27  DEVAADR32  /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)
description of BF10-11 should be changed to:
BF10-11 Slot 0 reserved

description of BF26-27 should be changed to:
BF26-27 /RAM

Under the "MACHINE IDENTIFICATION BYTE", the second column of numbers should read:

\[ \begin{array}{c}
0... \\
0... \\
0... \\
1... \\
1... \\
1...
\end{array} \]

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,A$2000
CALL -151
address*:40
3D0G
BSAVE PRODOS,TSYS,A$2000
LOCK PRODOS

*"address" varies with the version of ProDOS, as follows:

<table>
<thead>
<tr>
<th>ProDOS Version</th>
<th>address</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0.1</td>
<td>520D</td>
</tr>
<tr>
<td>1.0.2</td>
<td>52CD</td>
</tr>
<tr>
<td>1.1.1</td>
<td>56E3</td>
</tr>
</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>
BSLEMS

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>
<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

QUALITY SOFTWARE