SUPPLEMENT TO

Beneath Apple ProDOS

For ProDOS 8, Versions 1.2 and 1.3

by Don D. Worth and Pieter M. Lechner

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

Beneath Apple ProDOS
by Don Worth & Pieter Lechner
$19.95

Supplement to Beneath Apple ProDOS for Versions 1.0.1, 1.0.2
by Don Worth & Pieter Lechner
$10.00

Supplement to Beneath Apple ProDOS for Version 1.1.1
by Don Worth & Pieter Lechner
$12.50

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

"Apple" is a registered trademark of Apple Computer, Inc. This manual was not prepared nor reviewed by Apple Computer, Inc., and present any endorsement, official or otherwise, by Apple Computer, Inc.
## CONTENTS

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

### PRODOS 8, VERSIONS 1.2 AND 1.3

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

### BASIC.SYSTEM, VERSION 1.1

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

### DISK II BOOT ROM

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

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

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

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

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

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

UNDERSTANDING THE LISTINGS

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

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

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

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

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

   "P8"
   32 BLOCK FILE
   (31 data blocks plus one index block)
   L$3C7D

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

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

   I------------------I$FFFF
   I------------------I$FF9B
   I------------------I$FF00
   I------------------I$FFFF

4. Final moves:
   FUNCTION FROM TO LENGTH
   Clock code 5100* D742 7D
   QUIT code 5900 D10** 300
   RAM drive...

*5600* I IGS **BANK2
***AUX....MEMORY

---

(run location)

---

MLI DATA AREA

---

DISK II DRIVER

---

SYSTEM GLOBAL PAGE

---

IIIGS CLOCK CODE

---

QUIT CODE

---

DISK II DRIVER

---

IRQ HANDLER

---

CLOCK CODE

---

SYSTEM GLOBAL PAGE

---

MLI

(load location)

---

RELOCATOR

---
Beneath Apple ProDOS Software

ProDOS Loader -- V.1.2

** ADDRESSES **

---

** 0000 MODULE START ****

---

** 0001 MAIN ENTRY ****

---

** 0002 ENTRY POINT FOR (01 MEANS BOOT ROUTINE Follows)

---

** 0003 ALWAYS TAKEN (A0) DURING a 5.25" FLOPPY BOOT)

---

** 0004 JUMP TO APPLE ///c>

---

** 0005 SAVE SLOT 16

---

** 0006 MAKE $C6 FROM $C0 /// EXECUTES A HARMLESS

---

** 0007 AND SAVE AT $49 /// EXECUTES A HARMLESS

---

** 0008 CALL ROM SECTOR $38, X, THEN DOES NOT BRANCH

---

** 0009 CHECK $CFF

---

** 0010 CALL $A12 // THIS APPLE /// INFORMATION.

---

** 0011 GOSUB BOTH SECTORS

---

** 0012 GOSUB BOTH SECTORS

---

** 0013 STORE ON PARM ///

---

** 0014 SKIP SECTOR 1 ///

---

** 0015 DUMMY UP $C5C

---

** 0016 AND CALL ROM SECTOR $A12 //

---

** 0017 LOAD //

---

** 0018 LOGIC >>A132

---

** 0019 ENTIRE LOAD /// NEXT?

---

** 0020 CURRENT TRACK 1

---

** 0021 $48/49 --> $C6

---

** 0022 COPY A PORTION

---

** 0023 TO MY BLOCK READ IN ROM BOOT

---

** 0024 FROM $A97 TO $A97

---

** 0025 MODIFY SOME BLOCK

---

** 0026 TO SUIT MY FLOPPY ///

---

** 0027 ENTER OF LOADER? >>0831

---

** 0028 (0) GOSUB SEC 2).

---

** 0029 AS RETURN ADDRESS

---

** 0030 ProDOS *************

---

** 0031 IN MEMORY NOW)

---

** 0032 ZERO

---

** 0033 OF DISKETTE BOOT ROM

---

** 0034 EXT SUBROUTINE (0994)

---

** 0035 ACCESS IN THE COPIED CODE (091D)

---

** 0036 HANDLING TASTES (0924)

---

** 0037 ProDOS Loader -- V.1.2

---

** ADDR DESCRIPTION

---

** 0800 NEXT OBJECT ADDR: 0800

---

** 1.2 -- 6 SEP 86

---

** 1.3 -- 2 DEC 86

---

** ADDER IS STILL THE SAME AS IT

---

** 0000 SIGNATURE BYTE ///

---

** A $65 IS STORED // APPD

---

** THIS CODE (BLC

---

** ROM JUMPS TO

---

** $800 ON AN API

---

** THUS AN APPLE

---

** NEXT OBJECT ADDR: 0800

---

** ON CARRY, AND

---

** APPLE II). MA

---

** PROVIDING US

---

** 0010 SCREEN Prints

---

** A

---

** 0011 ENTRY POINT FOR

---

** 0012 ALWAYS TAKEN (A0)

---

** 0013 JUMP TO APPLE ///c>

---

** 0014 SAVE SLOT 16

---

** 0015 MAKE $C6 FROM $C0 /// EXECUTES A HARMLESS

---

** 0016 AND SAVE AT $49 /// EXECUTES A HARMLESS

---

** 0017 CALL ROM SECTOR $38, X, THEN DOES NOT BRANCH

---

** 0018 CHECK $CFF

---

** 0019 CALL $A12 // THIS APPLE /// INFORMATION.

---

** 0020 GOSUB BOTH SECTORS

---

** 0021 GOSUB BOTH SECTORS

---

** 0022 STORE ON PARM ///

---

** 0023 SKIP SECTOR 1 ///

---

** 0024 DUMMY UP $C5C

---

** 0025 AND CALL ROM SECTOR $A12 //

---

** 0026 LOAD //

---

** 0027 LOGIC >>A132

---

** 0028 ENTIRE LOAD /// NEXT?

---

** 0029 CURRENT TRACK 1

---

** 0030 $48/49 --> $C6

---

** 0031 COPY A PORTION

---

** 0032 TO MY BLOCK READ IN ROM BOOT

---

** 0033 FROM $A97 TO $A97

---

** 0034 MODIFY SOME BLOCK

---

** 0035 TO SUIT MY FLOPPY ///

---

** 0036 ENTER OF LOADER? >>0831

---

** 0037 (0) GOSUB SEC 2).

---

** 0038 AS RETURN ADDRESS

---

** ProDOS *************

---

** IN MEMORY NOW)

---

** ZERO

---

** OF DISKETTE BOOT ROM

---

** EXT SUBROUTINE (0994)

---

** ACCESS IN THE COPIED CODE (091D)

---

** HANDLING TASTES (0924)

---

** ProDOS Loader -- V.1.2

---

** ADDR DESCRIPTION

---

** 0800 NEXT OBJECT ADDR: 0800

---

** 1.2 -- 6 SEP 86

---

** 1.3 -- 2 DEC 86

---

** ADDER IS STILL THE SAME AS IT

---

** 0000 SIGNATURE BYTE ///

---

** A $65 IS STORED // APPD

---

** THIS CODE (BLC

---

** ROM JUMPS TO

---

** $800 ON AN API

---

** THUS AN APPLE

---

** NEXT OBJECT ADDR: 0800

---

** ON CARRY, AND

---

** APPLE II). MA

---

** PROVIDING US

---

** SCREEN Prints

---

** A

---

** ENTRY POINT FOR

---

** ALWAYS TAKEN (A0)

---

** JUMP TO APPLE ///c>

---

** SAVE SLOT 16

---

** MAKE $C6 FROM $C0 /// EXECUTES A HARMLESS

---

** AND SAVE AT $49 /// EXECUTES A HARMLESS

---

** CALL ROM SECTOR $38, X, THEN DOES NOT BRANCH

---

** CHECK $CFF

---

** CALL $A12 // THIS APPLE /// INFORMATION.

---

** GOSUB BOTH SECTORS

---

** GOSUB BOTH SECTORS

---

** STORE ON PARM ///

---

** SKIP SECTOR 1 ///

---

** DUMMY UP $C5C

---

** AND CALL ROM SECTOR $A12 //

---

** LOAD //

---

** LOGIC >>A132

---

** ENTIRE LOAD /// NEXT?

---

** CURRENT TRACK 1

---

** $48/49 --> $C6

---

** COPY A PORTION

---

** TO MY BLOCK READ IN ROM BOOT

---

** FROM $A97 TO $A97

---

** MODIFY SOME BLOCK

---

** TO SUIT MY FLOPPY ///

---

** ENTER OF LOADER? >>0831

---

** (0) GOSUB SEC 2).

---

** AS RETURN ADDRESS

---

** ProDOS *************

---

** IN MEMORY NOW)

---

** ZERO

---

** OF DISKETTE BOOT ROM

---

** EXT SUBROUTINE (0994)

---

** ACCESS IN THE COPIED CODE (091D)

---

** HANDLING TASTES (0924)
ProDOS Loader -- V1.2 -- 6 SEP 86

NEXT OBJECT ADDR: 084C

084C AND COPY SECTOR READ SUBROUTINE EXIT CODE (092B)
084F TO $A7F TO $A85 (0A7F)
0855 $48/49 --> DISKETTE BLOCK READER SUBRTN
0859 AT $0000
085B ---
085D LEGAL DISK ROM?
085F NO, ERROR >>0890
0861 STORE LSB OF BLOCK READER
0863 STORE ZEROS IN SEVERAL THINGS
086E COMMAND = 1 (READ BLOCK)
0871 BLOCK NUMBER = 2 (VOL DIRECTORY)
0875 $60/61 --> $C000 (BUFFEN)
0877 $4A/4B --> $C000 (FIRST ENTRY)
0879 READ VOLUME DIRECTORY BLOCKS <0912>
087C ERROR? >>08E6
087E MOVE UP TWO PAGES IN MEMORY
0882 NEXT BLOCK NUMBER
0886 NOW AT BLOCK 62
0888 NO, GO READ NEXT ONE >>0879
088A YES, CHECK LINK FOR VALIDITY ($00)
088D IT SHOULD BE ZERO FOR VOL DIR ($C01)
0890 BAD VOLUME DIR IF NOT ZERO >>08FF
0892 NO, INDEX PAST LINK AND VOL HDR
0894 AND BEGIN >>0898
0896 IF ALREADY PROCESSING, USE ENTRY LSB
0899 ---
089B ADD ENTRY LENGTH TO FIND NEXT ENTRY ($023)
089D STILL IN SAME PAGE? >>08AC
089F NO, BUMP ENTRY MSB
08A3 IS IT ODD? (SECOND PAGE OF A BLOCK?)
08A4 YES... >>08AC
08A6 NO, JUST FINISHED LAST BLOCK?
08A8 YES, ERROR -- FILE NOT FOUND >>08FF
08AA ELSE, START JUST PAST LINKS
08AC UPDATE LSB OF ENTRY POINTER
08AE GET NAME LENGTH ($002)
08B1 MASK OFF STORAGE TYPE
08B4 COMPARE NAME WITH "PRODOS"
08B9 NOT A MATCH? >>0896
08BE IF NAME MATCHES, IS IT A SAPLING FILE?
08C2 IF NOT, I CAN'T HANDLE IT >>08FF
08C6 GET FILE TYPE
08CB SHOULD BE A PRODOS SYS FILE
08CA IF NOT, I GIVE UP >>08FF
08CD ALL IS WELL, COPY KEY BLOCK NUMBER
08CF TO $46/47
08D6 $4A/4B AND $60/61 --> $1E00
08DB (BUFFER TO HOLD KEY BLOCK)
08E1 $4C/4D --> $1F00 (SECOND PAGE)
08E3 READ A BLOCK <0912>
08E6 ERROR? >>08FF

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

NEXT OBJECT ADDR: 08EA

08EA BUMP TO NEXT BLOCK BUFFER
08EB $4E OFFSET INTO INDEX
08F0 GET NEXT BLOCK NUMBER FROM INDEX BLOCK
08F8 BLOCK NUMBER = 07 (ENTRY FOR INDEX BLOCK
08FA NOT YET, READ A 'BLOCK OF FILE'
08FC ELSE, JUMP TO RELOCATE AT $2000 >>2000
08FF ERROR JUMP >>093F

0902 *********** KERNEL NAME ***********
0902 LENGTH OF KERNEL'S NAME
0903 'PRODOS': " PRODOS" (KERNEL NAME)

0912 *********** COPY BLOCK READ BUFFER PTR ***********
0912 COPY $60/61 --> $44/45
0914 (BLOCK READ BUFFER PTR)
091A THEN GO TO BLOCK 1 (GUTTER)

091D *********** ROM SECTOR READ SUBROUTINE ***********
091D OFFSETS INTO ROM AND OFFSETS ***********
091D TO BRANCH BETWEEN SECTOR READ SUBROUTINE
091D BE CHANGED FOR CHANGES WHICH NEED TO
091D SERV'S PURPOSES

091D ********** NEW BRANCH **********

0924 ---

092B *********** SECTOR READ SUBROUTINE ***********
092B COPY TO END OF DISKETTE SECTOR READ CODE

092D GET SLOT*16
092D AND EXIT NORMALLY
092E RETURN
092F RESTART BLOCK READ

0932 *********** APPLE /// BEGIN CODE ***********

A132 THIS IS A132 WHEN BDC
0932 MAKE IT LOOK LIKE A132 ON APPLE ///
093B LOAD IN BLOCK 1 (WE CAN FROM $A000)
093C GO TO APPLE /// BLOCK READ ROUTINE >>F479
Beneath Apple ProDOS Supplement

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

ADDR DESCRIPTION/CONTENTS

093F ************ ERROR HANDLER ***************

093F HOME CURSOR/CLEAR SCREEN <FC50>
0944 COPY "UNABLE TO LOAD PRODOS" MESSAGE (0950)
0947 TO SCREEN (05AE)
094D THEN GO TO SLEEP FOREVER >>094D

0950 ---
0958 *** 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 (C0BH)
097A ---
097C DELAY LONG ENOUGH FOR ARM TO MOVE
0983 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 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 <09BC>
09AD NEXT PAGE
09B1 SEKW TO NEXT SECTOR
09B5 READ SECOND SECTOR OF BLOCK <09BC>
09B8 THEN TURN MOTOR OFF AND EXIT (C083)
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

... of copied routine 'DESCRIPTIONS')
... boot firmware'...
Beneath Apple ProDOS Supplement

ProDOS Relocator -- v1.2 -- 6 SEP 86

MODULE STARTING ADDRESS

* ProDOS Relocator
* Loaded as the first
* Portion of the ProDOS
* Image at $2000
* Version 1.2 -- 6 SEP 86

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

04B8 SCREEN BUFFER ROW 10
05A9 SCREEN BUFFER ROW 12
05AD SCREEN BUFFER ROW 12
06D6 SCREEN BUFFER ROW 14
07AB SCREEN BUFFER ROW 16
07AD SCREEN BUFFER ROW 16
07D0 SCREEN BUFFER ROW 24

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

0800 ENTRY OF INTERP LOADER
0C00 VOLUME DIRECTORY BUFFER
0C23 ENTRY LENGTH
0E94 VOLUME HDR, VOLUME NAME
0E22 VOLUME HDR, ACCESS-TOTAL BLOCKS
2000 START OF SYSTEM PROGRAMS
2C00 RAMDRIVE DEVICE DRIVER LOAD ADDRESS
2A60 DIFFERENCE OF RAMDRIVE LOAD AND RUN LOCATIONS
BFFF TOP OF 4K RAM

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

BF00 ENTRY POINT FOR MLI
BF03 QUIT VECTOR
BF06 DATE/TIME
BF10 DEVICE HANDLER TABLES
BF30 LAST DEVICE USED
BF31 NUMBER OF ACTIVE DISK DEVICES
BF32 ACTIVE DISKS SEARCH LIST
BF98 MACHINE TYPE FLAGS
BF99 SLOT WHICH CONTAINS CARDS WITH ROM
BFFF MLI VERSION NUMBER

********* I/O PORT ADDRESSES **********

C000 80 STORE OFF
C001 80 STORE ON
C002 READ MAIN RAM
C003 READ AUX RAM
C004 WRITE MAIN RAM
C005 WRITE AUX RAM
C008 MAIN STACK/ZERO PAGE
C009 ALTERNATE STACK/ZERO PAGE
C00A INTERNAL SLOT 3 ROM
C00B PERIPHERAL SLOT 3 ROM
C00C 80 COLUMN DISPLAY OFF
C018 READ 80STORE SWITCH

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

000A AUTOSTART ROM CHECKSUM POINTER
000B
000C CONFIGURATION BYTE (MACHID TO BE)
0010 GENERAL PURPOSE POINTER
0011
0012 DISK TYPE (0=DISK II, 4=PROFILE)
0013 AND INPUT RELOC RANGE POINTER
0014 VOL DIR ENTRY POINT FOR RELOCATOR
0015 AND OUTPUT RANGE PTR
0016 LENGTH OF RELOCATION RANGE
0017
0018 INPUT RELOCATION RANGE POINTER
0019
001A END OF INPUT RANGE
001B
003C GENERAL PURPOSE POINTER
003D GENERAL PURPOSE POINTER
003E GENERAL PURPOSE POINTER
003F
0040 RAMDRIVE OUTPUT POINTER
0041
0042 VARIOUS USES: PARM TO AUXMOVE,
0043 UNIT/SLOT PASSED TO RELOCATOR
0046 BLOCK NUMBER TO RAMDRIVE
0047

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

0080 MACHID BUILD SUBRNT FOR 128K
0101 SAVE AUX STACK POINTER (IN AUX STACK)
0280 GENERAL PURPOSE BUFFER
0281 BUFFER+1
<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>C030</td>
<td>SPEAKER</td>
</tr>
<tr>
<td>C054</td>
<td>USE MAIN MEMORY PART OF 80-COL CARD</td>
</tr>
<tr>
<td>C055</td>
<td>USE AUX MEMORY PART OF 80-COL CARD</td>
</tr>
<tr>
<td>C060</td>
<td>IIGS STATEREG STATUS BYTE</td>
</tr>
<tr>
<td>C061</td>
<td>WRITE-ENABLE HIGH RAM</td>
</tr>
<tr>
<td>C082</td>
<td>MOTHERBOARD ROM READ ENABLE</td>
</tr>
<tr>
<td>C083</td>
<td>READ/WRITE RAM 2ND 4K BANK</td>
</tr>
<tr>
<td>C088</td>
<td>READ/WRITE RAM 1ST 4K BANK</td>
</tr>
<tr>
<td></td>
<td>********** INTERNAL CROM ADDRESSES **********</td>
</tr>
<tr>
<td>C311</td>
<td>MOVE TO/FROM AUXMEM SUBROUTINE</td>
</tr>
<tr>
<td>C314</td>
<td>TRANSFER TO/FROM AUXMEM SUBROUTINE</td>
</tr>
<tr>
<td></td>
<td>********** SLOT ROM ADDRESSES **********</td>
</tr>
<tr>
<td>C305</td>
<td>SLOT3 I.D. BYTE</td>
</tr>
<tr>
<td>C307</td>
<td>SLOT3 I.D. BYTE</td>
</tr>
<tr>
<td>C309</td>
<td>SLOT3 I.D. BYTE</td>
</tr>
<tr>
<td>C30C</td>
<td>SLOT3 I.D. BYTE</td>
</tr>
<tr>
<td>C30F</td>
<td>SLOT3 I.D. BYTE</td>
</tr>
<tr>
<td>CFFF</td>
<td>RESET I/O CARD ROM</td>
</tr>
<tr>
<td></td>
<td>********** PRODOS ADDRESSES **********</td>
</tr>
<tr>
<td>D000</td>
<td>START OF QTICODE MEMORY AREA (BANK2)</td>
</tr>
<tr>
<td>DFF1</td>
<td>ENHANCED ROM FLAG</td>
</tr>
<tr>
<td>FDB0</td>
<td>VERSION NUMBER FOR SUBDIRECTORIES</td>
</tr>
<tr>
<td>FEFF</td>
<td>GS VIDEO FLAG</td>
</tr>
<tr>
<td>FF00</td>
<td>RAMDRIVE CALLER ADDRESS</td>
</tr>
<tr>
<td></td>
<td>********** MONITOR ROM **********************</td>
</tr>
<tr>
<td>F81E</td>
<td>PADDLE READ SUBROUTINE</td>
</tr>
<tr>
<td>F82F</td>
<td>MONITOR INIT ROUTINE</td>
</tr>
<tr>
<td>F8B3</td>
<td>ROM VERSION BYTE</td>
</tr>
<tr>
<td>F8C0</td>
<td>SECONDARY VERSION BYTE (0-3)</td>
</tr>
<tr>
<td>FC5B</td>
<td>CLEAR SCREEN</td>
</tr>
<tr>
<td>FEBF</td>
<td>THIS ROUTINE CHECKS FOR IIGS</td>
</tr>
<tr>
<td>FEB4</td>
<td>SET NORMAL VIDEO</td>
</tr>
<tr>
<td>FEB9</td>
<td>INI#</td>
</tr>
<tr>
<td>FEB3</td>
<td>PR#0</td>
</tr>
</tbody>
</table>

2000 ************ PRODOS RELOCATOR MAIN ENTRY *************

2000 JUMP OVER PQUIT ENTRY >>2006
2003 SET FLAG INDICATING PQUIT ENTRY (IIGS) (21D1)
2006 STORE SLOT IN MLI ONLINE PARM
2008 PRINT "APPLE II PRODOS..." (25B1)
200E SET UP FOR COMMON MOVES (226E)
2014 RELOCATE SOME ROUTINES & DATA TO LOW MEMORY <2880>

2017 RELOCATION ERROR >>203C
201D BE SURE 4K OF MAIN MEMORY EXISTS (BFFF)
2024 IF NOT, ERROR >>204E
2029 MAKE DOUBLE SURE (BFFF)
202C ERROR THIS TIME >>204E
202E SELECT MOTHERBOARD ROMS (C082)
2031 DETERMINE MACHINE TYPE <21F7>
2036 PICK UP CONFIGURATION BYTE
2038 64K OR MORE MEMORY?
203A YES, WE HAVE 64K RAM >>203F
203C ERROR. MUST HAVE 64K OR MORE!! >>2227

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

203F SET UP FOR MLI MOVE (2270)
2045 COPY/RELOCATE PRODOS ITSELF <20B0>
2048 GET PRODOS VERSION NUMBER (BFFF)
204B AND PUT IT IN MLI DATA AREA. (FDDD)
204E RELOCATION ERROR! >>2BAC
2050 ENABLE MOTHERBOARD ROMS AGAIN (C082)
2053 CHECK ROM I.D. BYTE (FB03)
2056 APPLE //e FAMILY?
2058 NO, LEAVE I.D. BYTE AS IS >>2B5E
205C TEST ANOTHER ROM I.D. BYTE (FBC4)
205F SAVE BIT TEST RESULTS
2060 GET MACHID
2062 STRIP BITS THAT IDENTIFY MODEL
2067 IT'S A //e IF BITS 6 & 7 ARE HIGH >>2075
2069 ---
206A EITHER A //c OR A FUTURE SYSTEM
206C CHECK HIGH BITS OF FBC4 AGAIN
206D BIT 7 ON >>2073
206F YES, FUTURE SYSTEM.
2073 IF BIT 6 ON, IT'S A FUTURE SYSTEM. >>2077
2075 ---
2077 REPLACE UPDATED MACHID
207D LOOK AT ROM. THIS A IIGS? <F1F7>
2080 NO, CARRY STILL SET. >>208E
2082 YES, SET IIGS FLAG. (2278)
2085 ENTER FROM PQUIT? (21D1)
2088 YES, THIS IS NOT INITIAL BOOT. >>208E
2090 NOW SET OS BOOT TO ZERO, INDICATING THAT PRODOSB
2093 WAS THE OPERATING SYSTEM INITIALLY BOOTED.
209A G516 INSTRUCTIONS: STA $100BB
209B COPY BOOT DEVICE ID TO READ BLOCK PARM (2262)
209D AND AS LAST DEVICE USED (BF30)
2097 DETERMINE PERIPHERAL CARD CONFIGURATION <265F>
2099 BOOT DEVICE TO... (2269)
209D GLOBAL PAGE LAST DEVICE USED (WF30)
20A0 ENABLE READ/WRITE HIGH RAM, BANK 1 <2518>
20A9 COPY CLOCK CODE TO DEVICE DRIVER AREA <2880>
<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>20AC</td>
<td>ERROR: 20DA</td>
</tr>
<tr>
<td>20AE</td>
<td>CHECK MACHINE TYPE AGAIN (BF98)</td>
</tr>
<tr>
<td>20B1</td>
<td>GOT 32K OR MORE?</td>
</tr>
<tr>
<td>20B5</td>
<td>NO, &gt;&gt;20C3</td>
</tr>
<tr>
<td>20B7</td>
<td>YES, QUIT VECTOR --&gt; $FCA9</td>
</tr>
<tr>
<td>20C1</td>
<td>WRITE TO HIGH RAM (BANK2) (C083)</td>
</tr>
<tr>
<td>20CA</td>
<td>POINT TO QUIT CODE TABLE (2275)</td>
</tr>
<tr>
<td>20CD</td>
<td>MOVE QUIT CODE TO HIGH RAM &lt;2880&gt;</td>
</tr>
<tr>
<td>20DF</td>
<td>STORE QUIT VECTOR START PAGE (D000)</td>
</tr>
<tr>
<td>20D5</td>
<td>ENABLE READ/WRITE HIGH RAM, BANK 1 &lt;2518&gt;</td>
</tr>
<tr>
<td>20DA</td>
<td>RELOCATION ERROR &gt;&gt;2227</td>
</tr>
<tr>
<td>20DD</td>
<td>GET MACHID YET AGAIN (BF98)</td>
</tr>
<tr>
<td>20EE</td>
<td>128K2</td>
</tr>
<tr>
<td>20E4</td>
<td>NO, &gt;&gt;20C3</td>
</tr>
<tr>
<td>20E6</td>
<td>YES, SET UP AUX RAM</td>
</tr>
<tr>
<td>20E8</td>
<td>SAVE STATUS REG IN ACCUM</td>
</tr>
<tr>
<td>20EA</td>
<td>DISABLE INTERRUPTS</td>
</tr>
<tr>
<td>20EB</td>
<td>PREPARE TO WRITE TO AUX STACK AREA (C099)</td>
</tr>
<tr>
<td>20EE</td>
<td>AUX STACK POINTER SET TO $FF (0101)</td>
</tr>
<tr>
<td>20F0</td>
<td>BACK TO MAIN 2-PAGE, STACK (C088)</td>
</tr>
<tr>
<td>20F0</td>
<td>RESTORE STATUS REG</td>
</tr>
<tr>
<td>20F9</td>
<td>ESTABLISH RAM DRIVE IN AUX MEM &lt;2800&gt;</td>
</tr>
</tbody>
</table>

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

| 20FC | READ ROM (C081)         |
| 20F3 | GET ROM'S IRQ VECTOR (FFFF) |
| 2105 | ENABLE READ/WRITE HIGH RAM, BANK 1 <2518> |
| 2108 | CARRY CARRY IF IRQ VECTOR IN C3 ROM |
| 210A | FLAG FOR OLD ROM*       |
| 210C | IT'S NOT ROM >>2127     |
| 210E | SWITCH TO AUX STACK & HIGH RAM (C099) |
| 2110 | INITIALIZE AUX STACK POINTER TO $FF (0101) |
| 2116 | PUT IRQ VECTOR IN AUX HIGH RAM (FFFF) |
| 211C | BACK TO MAIN HIGH RAM & 2-PAGE (C086) |
| 211F | PUT IRQ VECTOR IN MAIN HIGH RAM (FFFF) |
| 2125 | INDICATE ENHANCED IRQ LOGIC ON BOARD |
| 2127 | STORE FLAG IN MLI DATA AREA (DFD1) |

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

| 212A |
| 212C | SET DS VIDEO FLAG=0 (FFFF) |
| 212F | THIS ERROR? (2278) |
| 2132 | NO, >>213A            |
| 2134 | YES, SET DS VIDEO FLAG (IN MLI) (FFFF) |
| 2137 | *أخلاق* SEARCHING SLOT 3. >>21A5 |
| 213A | ENABLE INTERNAL VIDEO FIRMWARE (C08A) |
| 213D | CHECK FOR ROM (BF99) |
| 2140 | IN SLOT 3... (213B) |

---

**Machine Type:**

- **ProDOS Relocator -- V1.2 -- 6 SEP 86**
- **NEXT OBJECT ADDR: 20AC**

**ADDRESS DESCRIPTION/CONTENT**

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>2144</td>
<td>THIS IS AN APPLE II?</td>
</tr>
<tr>
<td>2147</td>
<td>IT'S GOT 80-COL CAPABILITY &gt;&gt;21A5</td>
</tr>
<tr>
<td>214A</td>
<td>AT THE SLOT 3 ROM (C08B)</td>
</tr>
<tr>
<td>214F</td>
<td>AT Offset +$45 (C305)</td>
</tr>
<tr>
<td>2150</td>
<td>EXECUTE THE RULES! (C3FA)</td>
</tr>
<tr>
<td>2151</td>
<td>&lt;&lt; HAVE BIT INSTRUCTION AT $3C0A</td>
</tr>
<tr>
<td>2153</td>
<td>A byte, you followed the rules! &gt;&gt;21A5</td>
</tr>
<tr>
<td>2154</td>
<td>CONTROL BACK TO MOTHERBOARD ROM (C08A)</td>
</tr>
<tr>
<td>2158</td>
<td>ON 80-COL (C081)</td>
</tr>
<tr>
<td>215A</td>
<td>SET UP AUX MEM. (C085)</td>
</tr>
<tr>
<td>215D</td>
<td>A BYTE AT AUX $400 (D400)</td>
</tr>
<tr>
<td>2164</td>
<td>THE ACCUMULATOR LEFT</td>
</tr>
<tr>
<td>2166</td>
<td>DO THE SAME WITH $400 (D400)</td>
</tr>
<tr>
<td>2168</td>
<td>THE SAME? (D400)</td>
</tr>
<tr>
<td>216D</td>
<td>ALL THE SAME? (D400)</td>
</tr>
<tr>
<td>216F</td>
<td>OK, LEFT TO THE RIGHT</td>
</tr>
<tr>
<td>2171</td>
<td>ALL THE SAME? (D400)</td>
</tr>
<tr>
<td>2174</td>
<td>&lt;&lt; TO MAIN MEMORY (D054)</td>
</tr>
<tr>
<td>2176</td>
<td>ALL ON 80-COL (D000)</td>
</tr>
<tr>
<td>2178</td>
<td>GIVE OFF 80-COL (D000)</td>
</tr>
<tr>
<td>217B</td>
<td>GIVE OFF 80-COL MEMORY FOUND? &gt;&gt;21A5</td>
</tr>
<tr>
<td>217C</td>
<td>SO TURN OFF 80-COL FLAG (BF98)</td>
</tr>
<tr>
<td>2183</td>
<td>MACHINE I.D. BYTE.</td>
</tr>
<tr>
<td>2186</td>
<td>BRANCH &gt;&gt;21AA</td>
</tr>
<tr>
<td>2187</td>
<td>TURN ON 80-COL FLAG (BF98)</td>
</tr>
<tr>
<td>218A</td>
<td>=$I&gt; (D278)</td>
</tr>
<tr>
<td>218D</td>
<td>NO, &gt;&gt;21C8</td>
</tr>
<tr>
<td>218F</td>
<td>SET ENABLE IGS CLOCK DRIVER</td>
</tr>
<tr>
<td>2193</td>
<td>SET ADDRESS OF RELOCATE TABLE (D276)</td>
</tr>
<tr>
<td>2196</td>
<td>SET IGS CLOCK CODE (D277)</td>
</tr>
<tr>
<td>2199</td>
<td>PUT THE CODE AT $7742 &lt;20BD</td>
</tr>
<tr>
<td>219C</td>
<td>WACLOCK EXISTING MACHID (BF98)</td>
</tr>
<tr>
<td>219E</td>
<td>ER FROM PQUIT? (2211)</td>
</tr>
<tr>
<td>21A1</td>
<td>AT THE INITIAL.</td>
</tr>
<tr>
<td>21A5</td>
<td>ENABLE ROM FOR READ (C082)</td>
</tr>
<tr>
<td>21AD</td>
<td>THE SYSTEM FRAG. (4 = PRODOS 8 WAS INITAL BOOT)</td>
</tr>
<tr>
<td>21B0</td>
<td>NEXT OBJECT ADDR: 20AC</td>
</tr>
<tr>
<td>21B1</td>
<td>CHECK FOR ROM (BF99)</td>
</tr>
<tr>
<td>21B3</td>
<td>IN SLOT 3... (213B)</td>
</tr>
</tbody>
</table>

---

**Machine Type:**

- **ProDOS Relocator -- V1.2 -- 6 SEP 86**
- **NEXT OBJECT ADDR: 2144**
Beneath Apple ProDOS Supplement

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

**GET VOLUME LABEL**

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

**READ VOLUME DIRECTORY**

21F4 ---
21F5 $14/15 --> $C00
21F8 ---
220A BLOCK = 2 (VOLUME DIRECTORY) (226C)
2206 MLI: READ BLOCK <BF00>
220C ERROR? >>2227
2210 GET NEXT BLOCK NUMBER
2216 IF ZERO, END OF VOLUME DIRECTORY >>2224
2218 ADD TWO PAGES (ONE BLOCK) TO POINTER
221B AND STOP AT $1400 IN ANY CASE
2229 ELSE, READ NEXT BLOCK AS WELL >>21FB
2224 WHEN DONE, JUMP TO SYSTEM FILE LOADER >>0300

**ERROR HANDLER**

2227 ENABLE MOTHERBOARD RON's (C082)
222A CLEAR SCREEn <PC58>
222D PRINT "RELOCATION/CONFIG ERROR" (223B)
2238 THEN SLEEP FOREVER >>2238

**DATA**

2238 ---
223B ** RELOCATION / CONFIGURATION ERROR **

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

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

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

**ADDRESS OF COMMON MOVES**

2270 ADDRESS OF PRODOS RELOC
2272 ADDRESS OF THUNDERCLOCK RELOC TABLE
2274 ADDRESS OF QUIT CODE RELOC TABLE
2276 ADDRESS OF IIGS CLOCK & DRIVER RELOC TABLE
2278 IIGS FLAG. IF NON-ZERO, RELOC TABLE

2279 **RELOCATION TABLE**

+0:
+0 - ZERO BLOCK
+0 - RELOCATE OF MEMORY
+0 - RELOCATE
+0 - RELOCATE 2's ADDRESSES
+1/2: ADDR OF OUTPUT 2 BYTE ADDR
+3/4: LENGTH OF BLOCK
+5/6: ADDR OF INPUT 2 BLOCK
+7: Ranges top k in bytes
+8: START BLOCK (IF ANY)
+8-COUNT: END P. CORRECT FOR (-1)
+8-COUNT+COUNT: ADD PAGES

**COMMON MOVE V3 CORRECTION FACTOR**

2279 COPY (SYSTEM FILE LOADER) TABLE

227A TO $080
227C LEN=$213
227E FRM=$220B
2280 COPY (PAGE 3 IMAGE)
2281 TO =$30B
2283 LEN=$2A
2285 FRM=$24EE
2287 COPY (CHECKSUM)
2288 TO =$46A
228A LEN=$02
228C FRM=$14
228E COPY (CHECK FOR 80-COL)
228F TO =$90
2291 LEN=$46
2293 FRM=$256B
2295 END OF TABLE

**QUIT CODE**

2296 COPY (QUIT CODE)

2297 TO =$D100
2299 LEN=$380
229B FRM=$5900
229D END OF TABLE
### ProDOS Relocator -- V1.2 -- 6 SEP 86

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>229E</td>
<td>COPY (IRQ HANDLER)</td>
</tr>
<tr>
<td>229F</td>
<td>TO =$FF9B</td>
</tr>
<tr>
<td>22A1</td>
<td>LEN=65</td>
</tr>
<tr>
<td>22A3</td>
<td>FRM=$519B</td>
</tr>
<tr>
<td>22A5</td>
<td>COPY (SYSTEM GLOBAL PAGE)</td>
</tr>
<tr>
<td>22A6</td>
<td>TO =$BF00</td>
</tr>
<tr>
<td>22A8</td>
<td>LEN=100</td>
</tr>
<tr>
<td>22AA</td>
<td>FRM=$5000</td>
</tr>
<tr>
<td>22AC</td>
<td>ZERO (PRODOS KERNEL DATA AREA)</td>
</tr>
<tr>
<td>22AD</td>
<td>ADH=$D700</td>
</tr>
<tr>
<td>22AF</td>
<td>LEN=700</td>
</tr>
<tr>
<td>22B1</td>
<td>COPY (PRODOS KERNEL)</td>
</tr>
<tr>
<td>22B2</td>
<td>TO =$DE00</td>
</tr>
<tr>
<td>22B4</td>
<td>LEN=2100</td>
</tr>
<tr>
<td>22B6</td>
<td>FRM=$2F00</td>
</tr>
<tr>
<td>22B8</td>
<td>COPY (DISKETTE DRIVER)</td>
</tr>
<tr>
<td>22B9</td>
<td>TO =$0B00</td>
</tr>
<tr>
<td>22BB</td>
<td>LEN=780</td>
</tr>
<tr>
<td>22BD</td>
<td>FRM=$5200</td>
</tr>
<tr>
<td>22BF</td>
<td>END OF TABLE</td>
</tr>
</tbody>
</table>

### Thunderclock Table

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>22C0</td>
<td>COPY (THUNDERCLOCK CODE)</td>
</tr>
<tr>
<td>22C1</td>
<td>TO =$D742</td>
</tr>
<tr>
<td>22C3</td>
<td>LEN=7D</td>
</tr>
<tr>
<td>22C5</td>
<td>FRM=$5100</td>
</tr>
<tr>
<td>22C7</td>
<td>RELOCATE INSTRUCTIONS</td>
</tr>
<tr>
<td>22C8</td>
<td>TO =$D742</td>
</tr>
<tr>
<td>22CA</td>
<td>LEN=69</td>
</tr>
<tr>
<td>22CC</td>
<td>FRM=$D742</td>
</tr>
<tr>
<td>22CE</td>
<td>FOR ADDR=$C1XX-$D1XX</td>
</tr>
<tr>
<td>22D1</td>
<td>ADJUST BY=$S0</td>
</tr>
<tr>
<td>22D2</td>
<td>END OF TABLE</td>
</tr>
</tbody>
</table>

### IIGS Clock Table

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>22D3</td>
<td>COPY (IIGS CLOCK CODE)</td>
</tr>
<tr>
<td>22D4</td>
<td>TO =$D742</td>
</tr>
<tr>
<td>22D6</td>
<td>LEN=7D</td>
</tr>
<tr>
<td>22D8</td>
<td>FRM=$5C00</td>
</tr>
<tr>
<td>22DA</td>
<td>END OF TABLE</td>
</tr>
</tbody>
</table>

### System File Loader

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>22DB</td>
<td>MLQ: GET.FILE.INFO CALL &lt;BF00&gt;</td>
</tr>
<tr>
<td>22DF</td>
<td>FIRST SEE IF THERE IS AN ATINIT FILE</td>
</tr>
<tr>
<td>22E1</td>
<td>NO ERRORS &gt;&gt;22EA</td>
</tr>
<tr>
<td>22E2</td>
<td>IS ERROR &quot;FILE NOT FOUND&quot;?</td>
</tr>
<tr>
<td>22E4</td>
<td>YES, THAT'S OK. &gt;&gt;232C</td>
</tr>
<tr>
<td>22E7</td>
<td>NO, OTHER ERROR &gt;&gt;232F</td>
</tr>
<tr>
<td>22E8</td>
<td>GET FILE TYPE OF FILE FOUND (008D)</td>
</tr>
<tr>
<td>22E9</td>
<td>IS IT ATINIT FILE?</td>
</tr>
<tr>
<td>22E5</td>
<td>NO, ERROR &gt;&gt;232F</td>
</tr>
<tr>
<td>22E1</td>
<td>MLQ: OPEN CALL &lt;BF00&gt;</td>
</tr>
<tr>
<td>22E0</td>
<td>FILE DOESN'T OPEN &gt;&gt;232F</td>
</tr>
<tr>
<td>22E9</td>
<td>MLQ: GET EOB CALL &lt;BF00&gt;</td>
</tr>
<tr>
<td>22E7</td>
<td>CAN'T FIND EOF &gt;&gt;232F</td>
</tr>
<tr>
<td>2301</td>
<td>HIGH BYTE OF EOB ($A01)</td>
</tr>
<tr>
<td>2304</td>
<td>FILE TOO BIG &gt;&gt;232F</td>
</tr>
<tr>
<td>2306</td>
<td>MEDIUM BYTE OF EOB ($A00)</td>
</tr>
<tr>
<td>2309</td>
<td>MAX FILE SIZE IS $9800</td>
</tr>
<tr>
<td>230B</td>
<td>FILE TOO BIG &gt;&gt;232F</td>
</tr>
<tr>
<td>2310</td>
<td>PUT IN READ PARMS ($A07)</td>
</tr>
<tr>
<td>2313</td>
<td>GET LOW BYTE OF EOB ($9FF)</td>
</tr>
<tr>
<td>2316</td>
<td>MLQ: READ CALL &lt;BF00&gt;</td>
</tr>
<tr>
<td>231C</td>
<td>READ ERROR &gt;&gt;232F</td>
</tr>
<tr>
<td>231E</td>
<td>MLQ: CLOSE CALL &lt;BF00&gt;</td>
</tr>
<tr>
<td>2324</td>
<td>CLOSE ERROR &gt;&gt;232F</td>
</tr>
<tr>
<td>2326</td>
<td>READ ROM (C082)</td>
</tr>
<tr>
<td>2329</td>
<td>GO TO APPLICATION &lt;2000&gt;</td>
</tr>
<tr>
<td>232C</td>
<td>NOW LOOK FOR SYSTEM FILE &gt;&gt;$0BAB</td>
</tr>
<tr>
<td>232F</td>
<td>PRINT ERROR MESSAGE: &quot;UNABLE TO LOAD ATINIT FILE&quot; (233D)</td>
</tr>
<tr>
<td>233B</td>
<td>SLEEP FOREVER &gt;&gt;233B</td>
</tr>
<tr>
<td>233D</td>
<td>MSG LENGTH</td>
</tr>
<tr>
<td>233E</td>
<td>&quot;&quot; &quot;&quot; UNABLE TO LOAD ATINIT FILE &quot;&quot;</td>
</tr>
<tr>
<td>2364</td>
<td>GET_FILE INFO PARMS (FOR ATINIT FILE)</td>
</tr>
<tr>
<td>2365</td>
<td>LOCATED AT $889 WHEN EXECUTED</td>
</tr>
<tr>
<td>2366</td>
<td>PATHNAME ADDRESS</td>
</tr>
<tr>
<td>2368</td>
<td>FILE TYPE</td>
</tr>
<tr>
<td>2376</td>
<td>OPEN PARMS FOR ATINIT FILE</td>
</tr>
<tr>
<td>2377</td>
<td>AT $89B WHEN EXECUTED</td>
</tr>
<tr>
<td>2378</td>
<td>PATHNAME ADDRESS</td>
</tr>
<tr>
<td>2379</td>
<td>I/O BUFFER AT $1400</td>
</tr>
<tr>
<td>237B</td>
<td>REFNUM=1</td>
</tr>
</tbody>
</table>
Beneath Apple ProDOS Supplement

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

Next Object Addr: 237B

Addr Description/Contents

237C ---
237D "ATINIT"

2383 ********** Look For Normal System File **************
(This Code Executes At $8A08)

2383 $10/11 --> Volume Directory Entries
2385 INITIALLY AT $C000
2387 OFFSET BEYOND LINKS (+4)
2389 JUMP OVER NEXT INSTRUCTION >>238D

******** Scan Directory For System File *

238B E PICK UP LSB
238D ---
238E BUMP BY ENTRY LENGTH ($C23)
2391 UPDATE LSB
2393 PAGE OVERFLOW? >>23A7
2395 NO, ROOM FOR ONE MORE ENTRY? ($C23)
239A NO, CHECK MSB
239D START OF A BLOCK? >>23A9
239F NO, AT END OF DIRECTORY?
23A1 YES, FILE NOT FOUND In DIRECTORY >>23C1
23A3 NO, START NEW BLOCK At +4
23A5 AND UPDATE LSB
23A7 BUMP MSB
23A9 ---
23AD "System" File Type?
23AF No, Try Another. >>238B
23B2 INACTIVE ENTRY?
23B4 If So, Skip It >>23B8
23B8 SAVE NAME LENGTH At $280 ($280)
23BD Must Be At Least 8 Chars Long >>23B8
23BF JUMP AROUND Error Code >>23C3
23C1 Error - System File Not Found >>2430

23C3 ---
23C6 Is This ".System"?
23C8 (See $24E7) ($8A0C)
23CC No, Skip Entry >>23B8
23D0 Check All Characters In Name >>23C6

******** Load System File At $2000 ********

23D2 ---
23D4 ---
23D5 Copy Name To $281
23DC And To "Unable To Load" Msg ($982)
23E4 Add Blank At End Of Name
23E6 In Message ($9E3)

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

Next Object Addr: 23EA

Addr Description/Contents

238A NameLen + ERRORMSGLEN
238C SAVE At $24D1 ($9F6)
238F MLI: OPEN .SYSTEM FILE <BP00>
2395 OPEN Error >>243D
2397 MLI: GET EOF <BP00>
239D CAN'T GET EOF >>243D
23PF GET HIGH BYTE ($A01)
2402 File Too Big >>2457
2404 GET MEDIUM BYTE ($A00)
2407 Must Be Less Than $9000 Bytes
2409 Too Big. >>2437
240B Store Length In MLI READ PARMS ($A07)
240E GET LOW BYTE ($0FF)
2411 AND STORE In READ PARMS. ($A06)
2414 MLI: READ SYSTEM FILE INTO $2000 <BP00>
241A No Read Errors >>2422
241C Error, Bad Buffer?
241E Yes, File Was Too Large >>2457
2420 ELSE, "Unable To Load ..." >>243D
2422 MLI: CLOSE System File <BP00>
2428 CLOSE Error >>243D
242A Enable Motherboard ROM ($082)
242D AND Jump To Beginning Of File >>2000

2438 ********** Error Handlers ****************

243B ---
2432 PRINT "Unable To Find A .SYSTEM File" ($909)
243B THEN Go To sleep >>2462

243D Get Name Length ($9F6)
2440 Line Length
2443 Less Name Length ($9F6)
2446 Divided By 2
2447 Gives Offset To Center The Line ($9F6)
244B PRINT "Unable To Load ..." ($9D1)
2455 Go To sleep Forever >>2462

2457 ---
2459 PRINT "System Program Too Large" ($9B1)
2462 Go To sleep Forever >>2462

2464 ********** Data Area **********

2464 "=" unable To Find A .SYSTEM File ":
246C "=" System Program Too Large
246C "=" Unable To Load X.SYS

246D Name Len +13H (Len of MSG)
Beneath Apple ProDOS Supplement

ProDOS Relocator -- V1.2 -- 6

SEP 86 NEXT OBJECT ADDR: 24D1

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>24D2</td>
<td>MLI: OPEN PARM LIST</td>
</tr>
<tr>
<td>24D3</td>
<td>PATHNAME IS AT $280</td>
</tr>
<tr>
<td>24D5</td>
<td>I/O BUFFER AT $1400</td>
</tr>
<tr>
<td>24D7</td>
<td>REFNUM=1 F (AT $9FD)</td>
</tr>
<tr>
<td>24D8</td>
<td>MLI: GET EOF PARM LIST</td>
</tr>
<tr>
<td>24D9</td>
<td>REFNUM=1</td>
</tr>
<tr>
<td>24DD</td>
<td>MLI: READ LIST (AT $A)</td>
</tr>
<tr>
<td>24DE</td>
<td>REFNUM=1</td>
</tr>
<tr>
<td>24DF</td>
<td>READ INTO $2800</td>
</tr>
<tr>
<td>24EL</td>
<td>LENGTH (FROM EOF MARK 3A)</td>
</tr>
<tr>
<td>24E3</td>
<td>ACTUAL LENGTH READ 3A</td>
</tr>
<tr>
<td>24E5</td>
<td>MLI: CLOSE LIST (AT $A)</td>
</tr>
<tr>
<td>24E6</td>
<td>REFNUM=0, CLOSE ALL</td>
</tr>
<tr>
<td>24E7</td>
<td>'.SYSTEM'</td>
</tr>
</tbody>
</table>

FILE LOADER ****************************

ProDOS Relocator -- 1H RAM FOR READ/WRITE ****************************

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>2518</td>
<td>MACHINE ID 00. 0. APPLE II</td>
</tr>
<tr>
<td>251E</td>
<td>TWO READS OF 18</td>
</tr>
<tr>
<td>251F</td>
<td>RETURN 11. 0. APPLE II+</td>
</tr>
<tr>
<td>251F</td>
<td>DEP 0. 128K RAM</td>
</tr>
<tr>
<td>251F</td>
<td>AT FIRST</td>
</tr>
<tr>
<td>251F</td>
<td>2 (F83)</td>
</tr>
<tr>
<td>251F</td>
<td>&gt;254B</td>
</tr>
<tr>
<td>251F</td>
<td>&gt;&gt;254B</td>
</tr>
<tr>
<td>251F</td>
<td>IIGSB7</td>
</tr>
<tr>
<td>251F</td>
<td>ASSUME NOTHING</td>
</tr>
<tr>
<td>2523</td>
<td>GET A ROM BIT</td>
</tr>
<tr>
<td>2526</td>
<td>APPLE II? DHI &gt;&gt;2545</td>
</tr>
<tr>
<td>2528</td>
<td>YES, SET BIT</td>
</tr>
<tr>
<td>252A</td>
<td>NO,</td>
</tr>
<tr>
<td>252C</td>
<td>APPLE II OR</td>
</tr>
<tr>
<td>252E</td>
<td>YES, SET BIT</td>
</tr>
<tr>
<td>2530</td>
<td>NO,</td>
</tr>
<tr>
<td>2532</td>
<td>APPLE II= ?</td>
</tr>
<tr>
<td>2533</td>
<td>NO, STRANG R KNOWN MACHINE</td>
</tr>
<tr>
<td>2538</td>
<td>REALLY A II+ INSTR AT $A0</td>
</tr>
<tr>
<td>2538</td>
<td>&gt;&gt;2568</td>
</tr>
<tr>
<td>253B</td>
<td>YES &gt;&gt;254B</td>
</tr>
<tr>
<td>253F</td>
<td>/// EMULATION</td>
</tr>
<tr>
<td>2543</td>
<td>4 KABLE HIGH RAM (BANK1) &lt;2518&gt;</td>
</tr>
<tr>
<td>2544</td>
<td>RETURN</td>
</tr>
<tr>
<td>2545</td>
<td>PAGE EXISTS (0000)</td>
</tr>
<tr>
<td>2545</td>
<td>OTHERWISE, 1</td>
</tr>
<tr>
<td>2547</td>
<td>CREATE INVALID</td>
</tr>
<tr>
<td>2549</td>
<td>AND GO THERE (TO $00 TO ALLOW BANK SWITCH)</td>
</tr>
<tr>
<td>254B</td>
<td>UPDATE MACHINE</td>
</tr>
<tr>
<td>254D</td>
<td>READ/WRITE EM</td>
</tr>
<tr>
<td>2552</td>
<td>SEE IF HIGH 8</td>
</tr>
<tr>
<td>2554</td>
<td>IF PRESENT,</td>
</tr>
<tr>
<td>2556</td>
<td>OR II+ &gt;&gt;254A</td>
</tr>
<tr>
<td>256B</td>
<td>LATER. CHECK FOR 128K.</td>
</tr>
<tr>
<td>256B</td>
<td>CODE MOVES</td>
</tr>
<tr>
<td>256C</td>
<td>MORTY (C08)</td>
</tr>
<tr>
<td>256C</td>
<td>(ENTERED WITH 000)</td>
</tr>
<tr>
<td>256B</td>
<td>UPDATE MACHINE</td>
</tr>
<tr>
<td>256D</td>
<td>IF PLUS, IS IT</td>
</tr>
<tr>
<td>256F</td>
<td>IT'S A II= OR</td>
</tr>
<tr>
<td>2571</td>
<td>BANK TO AUX 2</td>
</tr>
<tr>
<td>2577</td>
<td>STORE A PATTERN</td>
</tr>
<tr>
<td>257A</td>
<td>AND AT $800</td>
</tr>
</tbody>
</table>

-----
2560 MAKE SURE PRODOS UNIT CONTINUES TO STAYS THERE
2562 IT DIDN'T ATTACH TO LEFT, SHIFTS THERE
2564 NOW SHIFT $$2564$$ TO LEFT (0C00)
2566 ARE THEY SHIFTED TO THE LEFT?
2568 NO, AUX RAM NOT SAME, SAME7 (0C00)
2570 DID $8800 MONOTHESE $2592
2572 NOW, DO WE HAVE TO COPPY ($8000)
2574 DON'T HAVE TO COPY $128K $2595
2576 --- 128K
2578 ---
2580 BANK BACK TO
2582 ONLY 64k>> RAM MEMORY ($004)
2584 AT 64K
2586 IN MACHID (E0)
2588 K
258A SET UP $A/B
258C IN MOTHERBOARD "APPLE II"
258E SET $A/B AND $B AT $E009
25883 SET SOMEWHERE. $128K
2590 SA NOW $8000 KONG >>25AF
2592 ---
2594 RETURN TO OLD

259A ************** DISPLAY LOAD MESSAGE ***************
259B CLICK SPEAKER
259C STORE IN MASS (1) 0
259E SET NORMAL LAY ($EF00)
253D CALL MONITOR VIDEO <FBF>
253F SET VIDEO FIRMWARE <FBF>
2540 SET KEYBO ROM >>259F
2546 OUT OF DECIMAL >>259G
2557 CLEAR SCREEN >>259H
255C PRINT "APPLE //" >>2599
25D7 PRINT "PRODOS 8" >>25FA
25FA "COPYRIGHT 1983-86"

255A ************** DATA AREA **********************
255B 'APPLE //'
255A3 'PRODOS 8'
2560 "COPYRIGHT 1983-86"
<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>26B9</td>
<td>SAVE DEVICE ADDRESS HIGH BYTE (265C)</td>
</tr>
<tr>
<td>26BC</td>
<td>ESTABLISH DEVICE DRIVER IN GLOBAL PAGE &lt;2814&gt;</td>
</tr>
<tr>
<td>26BF</td>
<td>ONLY ONE DRIVE?</td>
</tr>
<tr>
<td>26C8</td>
<td>YES, GO TO NEXT SLOT &gt;&gt;26DE</td>
</tr>
<tr>
<td>26E0</td>
<td>IF TWO DRIVES WERE ASSIGNED, MOVE THEM TO THE BOTTOM OF THE LIST IN REVERSE ORDER</td>
</tr>
<tr>
<td>26DE</td>
<td>---</td>
</tr>
<tr>
<td>26DF</td>
<td>CARRY IS NOW CLEAR IF A PRODOS STORAGE DEVICE WAS FOUND IN THIS SLOT. OTHERWISE, CARRY IS SET.</td>
</tr>
<tr>
<td>26E2</td>
<td>MOVE DOWN ONE SLOT</td>
</tr>
<tr>
<td>26E8</td>
<td>WE'VE DONE ALL SLOTS &gt;&gt;26ED</td>
</tr>
<tr>
<td>26EA</td>
<td>CHECK NEXT SLOT &gt;&gt;267E</td>
</tr>
<tr>
<td>26F1</td>
<td>STASHED ANY DEVICES AT BOTTOM OF LIST? (265D)</td>
</tr>
<tr>
<td>26F6</td>
<td>NO. &gt;&gt;271A</td>
</tr>
<tr>
<td>26F8</td>
<td>YES, MOVE THEM BACK</td>
</tr>
<tr>
<td>2700</td>
<td>IN REVERSE ORDER.</td>
</tr>
<tr>
<td>2716</td>
<td>DONE WHEN X=Y (265D)</td>
</tr>
<tr>
<td>2717</td>
<td>---</td>
</tr>
<tr>
<td>271C</td>
<td>START AT BOTTOM OF SEARCH LIST (BF31)</td>
</tr>
<tr>
<td>271F</td>
<td>GET A DEVICE FROM LIST (BF32)</td>
</tr>
<tr>
<td>2722</td>
<td>PUT IT ON THE STACK</td>
</tr>
<tr>
<td>2725</td>
<td>IS IT THE CURRENT SLOT? (BF30)</td>
</tr>
<tr>
<td>2729</td>
<td>NO. KEEP LOOKING &gt;&gt;272D</td>
</tr>
<tr>
<td>272B</td>
<td>YES, TAKE IT OFF THE STACK</td>
</tr>
<tr>
<td>272C</td>
<td>INDICATE CURRENT SLOT FOUND</td>
</tr>
<tr>
<td>272D</td>
<td>---</td>
</tr>
<tr>
<td>272E</td>
<td>MORE TO CHECK &gt;&gt;271F</td>
</tr>
<tr>
<td>2730</td>
<td>GET DEVICE COUNT (BF31)</td>
</tr>
<tr>
<td>2739</td>
<td>CURRENT SLOT NOT FOUND! &gt;&gt;274A</td>
</tr>
<tr>
<td>273E</td>
<td>PUT CURRENT DRIVE AT (BF30)</td>
</tr>
<tr>
<td>273F</td>
<td>BOTTOM OF SEARCH LIST (BF32)</td>
</tr>
<tr>
<td>2743</td>
<td>ONLY ONE DEVICE ON LIST &gt;&gt;2751</td>
</tr>
<tr>
<td>2740</td>
<td>ONLY ONE DRIVE ON BOOT SLOT &gt;&gt;274A</td>
</tr>
<tr>
<td>2745</td>
<td>CHANGE DRIVE NUMBER</td>
</tr>
<tr>
<td>2744</td>
<td>STORE OTHER DRIVE NEXT TO LAST (BF32)</td>
</tr>
<tr>
<td>274B</td>
<td>CURRENT SLOT ONLY SLOT ON LINE &gt;&gt;2751</td>
</tr>
<tr>
<td>274A</td>
<td>GET OTHER DEVICES</td>
</tr>
<tr>
<td>274B</td>
<td>MOVE THEM AHEAD OF CURRENT DRIVE (BF32)</td>
</tr>
<tr>
<td>274F</td>
<td>STILL MORE TO DO &gt;&gt;274A</td>
</tr>
<tr>
<td>2751</td>
<td>---</td>
</tr>
<tr>
<td>2754</td>
<td>DOESN'T SAY &quot;APPLE II&quot;, &gt;&gt;275A</td>
</tr>
<tr>
<td>2756</td>
<td>WE'RE HAPPY, STORE FINISHED MACHID (BF98)</td>
</tr>
<tr>
<td>2759</td>
<td>AND LEAVE</td>
</tr>
<tr>
<td>275A</td>
<td>unknown machine, so die horribly! &gt;&gt;2545</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>275D</td>
<td>PUT A DEVICE ON DEVICE LIST ___________________________</td>
</tr>
<tr>
<td>2761</td>
<td>COMBINE DSSS with III</td>
</tr>
<tr>
<td>2762</td>
<td>BUMP DEVICE COUNT BY ONE</td>
</tr>
<tr>
<td>2763</td>
<td>AND ADD DRIVE TO SYSTEM SEARCH LIST (BF32)</td>
</tr>
<tr>
<td>2766</td>
<td>ROLL LEFT ANTICIPATING ROLL RIGHT</td>
</tr>
<tr>
<td>2767</td>
<td>RETURN</td>
</tr>
<tr>
<td>2768</td>
<td>IDENTIFY I/O CARD ___________________________</td>
</tr>
<tr>
<td>2769</td>
<td>WE ALREADY FOUND ROM IN THIS SLOT &gt;&gt;27C9</td>
</tr>
<tr>
<td>276C</td>
<td>CHECK SIGNATURE ON CARD FOR THUNDERCLOCK</td>
</tr>
<tr>
<td>2771</td>
<td>NOT IT &gt;&gt;27BD</td>
</tr>
<tr>
<td>2777</td>
<td>THUNDERCLOCK, WHICH SLOT?</td>
</tr>
<tr>
<td>2779</td>
<td>SAVE SLOT NUMBER (LESS 1)</td>
</tr>
<tr>
<td>277B</td>
<td>IN CLOCK CODE RELOCATION TABLE (22D1)</td>
</tr>
<tr>
<td>2780</td>
<td>ENABLE CLOCK/CALENDAR JUMP IN GLOBALS (BF96)</td>
</tr>
<tr>
<td>2785</td>
<td>NO MACHID! &gt;&gt;2751</td>
</tr>
<tr>
<td>2787</td>
<td>INDICATE THAT A CLOCK IS PRESENT</td>
</tr>
<tr>
<td>2789</td>
<td>AND UPDATE MACHID</td>
</tr>
<tr>
<td>278B</td>
<td>GO MARK ROM IN THIS SLOT &gt;&gt;27C9</td>
</tr>
<tr>
<td>2790</td>
<td>CHECK FOR PASCAL 1.1 PROTOCOL</td>
</tr>
<tr>
<td>2791</td>
<td>$CA85 = $3B?</td>
</tr>
<tr>
<td>2793</td>
<td>$CA87 = $18?</td>
</tr>
<tr>
<td>2799</td>
<td>NO. &gt;&gt;27B8</td>
</tr>
<tr>
<td>27A1</td>
<td>$CA0B = $01?</td>
</tr>
<tr>
<td>27A3</td>
<td>NO, BAD SIGNATURE &gt;&gt;27B8</td>
</tr>
<tr>
<td>27A4</td>
<td>YEA, GET LEFT Wobble</td>
</tr>
<tr>
<td>27A8</td>
<td>$B COLUMN CARD?</td>
</tr>
<tr>
<td>27AC</td>
<td>NO, UNKNOWN CARD &gt;&gt;27B8</td>
</tr>
<tr>
<td>27B8</td>
<td>NO MACHID! &gt;&gt;2751</td>
</tr>
<tr>
<td>27B2</td>
<td>MARK $B COLUMN CARD PRESENT</td>
</tr>
<tr>
<td>27B4</td>
<td>AND UPDATE MACHID</td>
</tr>
<tr>
<td>27B6</td>
<td>GO MARK ROM IN THIS SLOT &gt;&gt;27C9</td>
</tr>
<tr>
<td>27BB</td>
<td>unknown card, check rom to</td>
</tr>
<tr>
<td>27BC</td>
<td>SEE IF IT WILL HOLD A VALUE</td>
</tr>
<tr>
<td>27C2</td>
<td>FOR SOME TIME.</td>
</tr>
<tr>
<td>27C9</td>
<td>WE FOUND ROM IN THIS SLOT</td>
</tr>
<tr>
<td>27CB</td>
<td>CONVERT SLOT NUMBER</td>
</tr>
<tr>
<td>27CE</td>
<td>TO A BIT POSITION (2752)</td>
</tr>
<tr>
<td>27D1</td>
<td>AND OR INTO SUBLTYP (BF99)</td>
</tr>
<tr>
<td>27D7</td>
<td>RETURN TO CALLER</td>
</tr>
</tbody>
</table>
27D8 ******** DATA AREA ***********************************************

2708 DISK II DEVICE DRIVER ENTRY POINT
270A DEVICE SIGNATURE FOR:
270C +0,+2,+4,+6 = THUNDERCLOCK
270E +1,+3,+5,+7 = DISK
2710 (+7 NOT CHECKED)

272E BIT POSITION TABLE FOR SLOTS
2735 (ALSO USED IN CHECKSUM CALCULS)

27EA ******** COMPUTE AUTOBOOT ROM CHECKSUM **********************

27EA --
27EA GET ZERO IN INDEX REGISTER (27E2)
27EE POINT TO $FBF9 ("APPLE II" IN ROM)
27F0 MAKE SURE UPPER CASE
27F5 UPDATE CHECKSUM (27E2)
27F8 PUT HIGH BIT IN CARRY (27E2)
27FC DO 8 BYTES IN ALL (27E5)
2800 ACCUM = $00
2805 ACCUM = $80
2807 TURN ON high BIT (27E2)
280A ADD A FUDGE FACTOR
280C ON W0! A CLONE! >>2811
280E PASSED THE TEST...RETURN WITH MACHID
2810 RETURN
2811 ELSE, RETURN WITH ZERO MACHID
2813 RETURN

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

2814 SAVE CARRY (NUMBER OF DRIVES)
2815 GET HIGH BYTE OF SLOT ADDRESS
2817 MAKE IT SLOT NUMBER
2819 TIMES 2
281A USE LATER IN Y-REG
281B NOW GET SLOT*16 IN ACCUM
281D NOW HAVE $000000 (DRIVE 1)
281E PUT DEVICE ID ON DEVICE LIST <275D>
2821 GET BACK CARRY (NUMBER OF DRIVES)
2822 ROLL CARRY INTO ACCUM
2823 ONLY ONE DRIVE. >>2829
2825 TWO DRIVES. BUMP DEVICE COUNT
2826 AND PUT SECOND DRIVE ON SEARCH LIST (BP32)
2829 STORE FINAL DEVICE COUNT (BP31)
282C SHIFT DRIVE INDICATOR BACK TO CARRY
282D PUT IN LOW BYTE OF DEVICE DRIVER ADDRESS (2658)
2830 PUT IN GLOBAL PAGE FOR DRIVE 1 (BP10)

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

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

2844 PUT LEFT NIBBLE OF STATUS BYTE IN $12 <288D>
2847 GET HIGH BYTE OF SLOT ROM ADDRESS
2849 STORE IT IN RELOC DATA AREA (265C)
284C GET PRODOS ENTRY, LOW BYTE (2658)
2850 ADD THREE TO GET SMARTPORT ENTRY
2852 POKE INTO SMARTPORT CALL (285C)
2858 POKE THE HIGH BYE, TOO. (285D)
285E SELF MODIFIED TO CALL THE SMARTPORT <0W0>
285E VIEW AT STATUS COMMAND.
285F PARMLIST AT $28A8
2861 GET NUMBER OF DEVICES ON LINE (2653)
2864 NONE ON LINE! >>28A8
2866 INDICATE IF DRIVE 2 EXISTS.
2868 PUT DRIVER ADDRESS IN GLOBAL PAGE <2814>
286D IS THIS SLOT 5?
286F NO. >>28A8
2871 SLOT 2 BEING USED BY STORAGE DEVICE? (2658)
2874 YES, TWO DRIVES IS ALL YOU GET! >>28A8
2878 GET NUMBER OF DEVICES AGAIN (2653)
2879 MORE THAN TWO DRIVES?
287B NO. >>28A8
287D SET CARRY IF DRIVE 4 EXISTS.
287F PUT THEM IN SLOT 2
2881 PUT DRIVER ADDRESS IN GLOBAL PAGE <2814>
288A GO PROCESS NEXT SLOT >>26DE

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

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

2896 *** CHECK FOR PRODOS STORAGE DEVICE ********************************

2898 RESET I/O CARD ROMS (CFFF)
289D CHECK 3 BYTES ON CONTROLLER ROM
28A2 ANTICIPATE FAILURE
28A3 NOT A PRODOS STORAGE DEVICE >>28A8
28A9 SUCCESS--THIS IS A PRODOS STORAGE DEVICE.
28A9 RETURN
86 NEXT OBJECT ADDR: 28AA

** ATTENTION CALL **********************

***** COMMAND LIST FOR 564

28AB 3 PARAMETERS
28AC 3 ETAL STATUS CALL
28AD 1 ET STATUS DATA AT $2652
28AE 1 STATUS CODE IS $80

***** RELOCATION ROUTINE ADDR

(X/Y REGS CONTAIN TAB

28AF 2 EA
28B0 16 PASS THE TABLE ADDRESS TO (C068)
28B1 17 PASS THE I/S ADDRESS TO (C069)
28B2 18 ROM BANK $0
28B3 19 TURN OFF SLOT ROM, ENDU
28B4 07
28B5 08 SET OPERATION CODE: ($08)
28B6 09 SET OPERATION CODE: ($09)
28B7 0A SET JOB ADDRESS $14/15 -> OUTPUT BLOCK
28B8 0B 0C/13 -> LENGTH
28B9 0D 0E/13 -> NEGATIVE LENGTH
28BA 0F CHECK OPERATION CODE
28BB 10 ZERO BLOCK -> 2941
28BC 11 $12/13 = $18/19
28BD 12 END OF INPUT Block
28BE 13 INPUT BLOCK
28BF 14 copy LOCATION OF TABLE
28C0 15 END OF RANGES
28C1 16 copy START PAGES TO TABLE
28C2 17 AND END PAGES
28C3 18 AND FINALLY, LOCATION
28C4 19 JUMP TO NEXT TABLE ENTRY
28C5 20 STORE OPERATION CODE
28C6 21 JUMP TO LOCATION INSTRUCT.

***** 2/3 - RELOCATE A

28C8 22 0 RELOCATE ADDRESS <296>
28C9 23 COPY BLOCK <2976>
28CA 24 IF CONTINUE IF ALL WENT
28CB 25 RETURN
28CC 26 JUMP TO ERROR EXIT

ProDOS Relocator -- V1.2 -- 6 SEP 86 NEXT OBJECT ADDR: 2938

293B ********** 4 - RELOCATE INSTRUCTIONS **************

293B 4 RELOCATE INSTRUCTIONS <29DF>
293C 5 AND THEN COPY BLOCK >>2938

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

2941 0 BUMP TABLE POINTER TO NEXT ENTRY <296B>
2942 1 GET NUMBER OF PAGES TO DO
2943 2 NO FULL PAGES? >>2956
2944 3 ZERO AN ENTIRE PAGE
2945 4 BUMP PAGE POINTER
2946 5 DECREMENT LENGTH
2947 6 GET LENGTH OF PARTIAL LAST PAGE
2948 7 NO PARTIAL PAGE? >>2962
2949 8 ZERO PARTIAL PAGE TOO
294A 9 DONE, GET NEXT TABLE ENTRY >>28BC

2955 ********** 1 - COPY BLOCK **********************

2955 1 BUMP TABLE POINTER <296B>
2956 2 COPY BLOCK >>2930

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

296B 0 ADD FINAL ENTRY INDEX...
296C 1 TO TABLE ENTRY ADDRESS
296D 2 RETURN

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

2976 ---

297A 4 INPTR < OUTPTR? >>2987
297B 5 NO, GREATER? >>29AA
297C 6 MSB'S ARE EQUAL, CHECK LSB'S ALSO
297D 7 EXIT IF EQUAL
297E 8 INPTR < OUTPTR, COPY LAST PAGES FIRST
297F 9 BUMP BOTH INPTR AND OUTPTR BY...
2980 10 LENGTH-1 TO POINT AT LAST BYTE
2981 11 START WITH SHORT LAST PAGE LENGTH
2982 12
2983 13 COPY BYTES BACKWARDS THROUGH MEMORY
2984 14 DROP ADDRESSES AND LENGTH BY 256
2985 15 AND CONTINUE UNTIL FINISHED >>2999
2986 16 RETURN
ProDOS Relocator -- V1.2 -- 6 SEP 86
NEXT OBJECT ADDR: 29A9

ADDR    DESCRIPTION/CONTENTS
--------- ---------------------------------------
29AA    INFTR > OUTFTR, COPY PAGES FORWARD
29AC    HOW MANY FULL PAGES LEFTY?
29AE    NONE? >> 29BF
29B0    COPY A FULL PAGE
29B7    AND BUMP ADDRESSES
29B8    DECREMENT LENGTH BY 256
29BD    AND DO ALL PAGES >> 29B8
29BF    GET LENGTH OF LAST PAGE
29C1    EVEN PAGE BOUNDARY? >> 29CC
29C3    NO, COPY SHORT LAST PAGE
29CC    RETURN

29CD    ********* ADDR/PAGE RELOCATE ****************************
29DF    ********* INSTRUCTIONS RELOCATE ****************************
29E1    GET 6582 CODE
29E3    COMPUTE INSTRUCTION LENGTH <2A3A>
29E6    INVALID CODE? >> 29F9
29E8    3 BYTE INSTRUCTIONS?
29EA    NO >> 29F3
29EC    YES, 3 BYTE ADDRESS TO CORRECT
29EE    RELOCATE ADDRESS <2A8B>
29F1    AND ADVANCE BY 3 BYTES
29F3    NEXT INSTRUCTION <2A27>
29F6    CONTINUE UNTIL FINISHED >> 29DF
29F8    RETURN

******* INVALID CODE *************
29F9    POP THE STACK
29FB    RETURN WITH POINTER TO BAD INSTRUC.
2A02    RETURN

2A03    ******** ERROR RETURN ****************************

ProDOS Relocator -- V1.2 -- 6 SEP 86
NEXT OBJECT ADDR: 2A03

ADDR    DESCRIPTION/CONTENTS
--------- ---------------------------------------
2A03    RETURN WITH POINTER
2A07    EXIT WITH ERROR CODE
2A0A    RETURN
2A0B    ********** RELOCATE ABSOLUTE ADDRESS ********************************************************
2A0B    GET PAGE NUMBER TO CHECK
2A0D    GET NUMBER OF RANGES (LESS ONE) (2A98)
2A10    IS IT PRIOR TO START OF THIS RANGE? (2A91)
2A13    YES? >> 2A1C
2A15    NO, IS IS AFTER END OF RANGE? (2A99)
2A18    NO >> 2A20
2A1C    ---
2A1D    CHECK EACH RANGE >> 2A16
2A1F    RETURN
2A20    ---
2A21    ADD FUDGE FACTOR TO ADDRESS (2A1A)
2A24    AND UPDATE IT
2A26    RETURN
2A27    ********** BUMP POINTER TO NEXT ADDR ******************************************************
2A27    ---
2A28    ADD LENGTH TO POINTER
2A2F    CHECK TO SEE IF WE ARE DONE
2A35    ---
2A39    RETURN
2A3A    ********** COMPUTE INSTRUCTION LENGTH ********************************************************
2A3A    A-REG CONTAINS CODE
2A3B    ISOLATE LAST TWO BYTES FOR LATER
2A40    USE LAST 6 BITS AS TABLE INDEX
2A42    GET BYTE WITH 4 LENGTHS IN IT (2A4F)
2A45    ---
2A46    USING TOP TWO BITS AS INDEX... >> 2A4C
2A48    SHIFT DOWN THE PROPER LENGTH
2A4C    AND ISOLATE IT IN A-REG
2A4E    RETURN
2A4F    ********** 6582 OP LENGTH TABLE ***********************************************************
each byte contains four 2 bit lengths
2A4F    ---
Beneath Apple IIe DOS Supplement

ProDOS Relocator -- v1.2 setting (C018)
ADDR DESCRIPTION/CONTENTS
3BD FORMAT COMMAND? ($3BC)
3CD LOGIC >> 2C4F

2C80 SAVE THE BIOS
2C84 FORCE RAM READ/WRITE RAMDRIVE ***************
2C89 COPY INPUT PARMS
2C9B TO AUX PAGE 3 (WANTED)
2C91 FIRST TIME IN OPE WE ACTUAL DIRECTORY
2C14 NO, SKIP FORMAT: Y BLOCK <0335>

****** FORMAT: Y BLOCK (<3E04)

2C10 YES, SAVE BLOCK
2C18 PAGES $0E AND $0F
2C19 ZERO THE DIRENTRY ($03C2)
2C1F COPY VOLUME NAME BYTE TO ZERO ($03C2)
2C22 TO VOLUME DIRECT
2C28 LAST BYTE IN VOLU ($3D6)
2C2A IS AN $FE $3D; TRY BLOCK ($0E22)
2C2D $FF TO ACCUM. COMMAND? ($3BC)
2C30 14 $FF'S TO BITMAP
2C36 SET FIRST BIT = CONTINUE WITH READ/WRITE ($3BC)
2C39 COPY $BYTES NUMBER ($3C1)
2C3B OF DIRECTORY DATA
2C3E TO VOLUME DIRECT: WRITE RAMDRIVE BLOCK ******
2C44 WAS THIS A FONT?
2C47 YES, DONE. >> 2C48 TO PAGE NUMBER ($3C1)
2C49 NO, SET FLAG 4, EXIT
2C4C RESTORE BLOCK & Menu

****** READ

<table>
<thead>
<tr>
<th>VOLUME BIT MAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>03BC</td>
</tr>
</tbody>
</table>

2C4F CONVERT BLOCK LOAD/WRITEX 0342
2C55 THIS PAGE IN ROM
2C57 YES > 2C63 RITE IN AUX HIGH RAM *****
2C59 NO, IS IT BLOCK??
2C5B NO > 2C60
2C5D YES, DUMMY UP AT <02E5>
2C60 ELSE, NORMAL EXECUTE STATUS

****** READ/WRITE

| INVOKE BANK 17 |

2C63 SAVE PAGE NUMBER (+C73)
2C64 FIND IT IN MEMORY (+C73)
2C67 REMEMBER READ/WRITE/AUX HIGH RAM >> 2C79
2C68 WRITING >> 2C68 HIGH RAM (C0B3)
2C6A GET SAVED PAGE IN CRT (C0B3)
2C6B DOES OPERATION IN
2C6D NO, USE BANK2 ?
2C6F YES, FORCE IT TO
2C71 AND USE BANK1 OF
2C73 USE BANK2 OF AUX
2C76 AND WRITE ENABLES
Beneath Apple ProDOS Supplement

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

---

ADDR DESCRIPTION/CONTENTS
---

2C79 SAVE PAGE NUMBER IN BLOCK (4) BETWEEN (03C9)
2C7C PRESERVE HIS BUFFER ADDR (03BF)
2CB6 AUX HIGH RAM.
2CB9 SELECT AUX HIGH RAM (C099)
2CBF USE RAMDRIVE BUFFER AS AN &L (03BF)
2CC8 AREA WHEN TRANSFERRING TO/FROM (03BF)
2CD Pretend that was caller's buffer
2CE0 AND SET UP POINTERS AGAIN <0
2CF4 COPY BLOCK TO OR FROM RAMDRIVE
2CFF THEN BACK TO MAIN ZERO PAGE RAM (RAM1) (C09B)
2C2A RESTORE CALLER'S BUFFER ADDRESS
2CA9 READING OR WRITING?
2CAF IF WRITING, DONE >>2C5
2CAE IF READING, WRITE ENABLE HIGH
2CB2 AND COPY RAMDRIVE BUFFER TO BUFFER
2CB5 THEN EXIT >>03DE
2CB8 IF WRITING, COPY HIS BLOCK TO BUFFER
2CBE THEN COPY RAMDRIVE BUFFER TO RAM DRIVES

---

2CBB COPY BLOCK IN MAIN
2CBB THIS ENTRY IS FOR THE RAMDRIVE
2CC3 THIS ENTR Y ASSUMES AUX MEM
2CC6 WRITING TO RAMDISK >>2DB SWING "JUMP"
2CC8 NO, WRITE TO MAIN 48K RAM (C098)
2CCB COPY BLOCK TO AUX MEM >> MAIN VALUE (03ED)
2CD7 WRITE TO AUX MEM AGAIN (C095)
2CD8 DONE (RETURN HERE AFTER FULL)
2CD9 --- ADDRESSES **************
2CE0 TO COPY MAIN MEM -- AUX MEM

---

2CE5 ********** SET BUFFER AND BLOCK TO READ (03C8)
2CEF GET COMMAND (03BD)
2CF6 READ OR WRITE?
2CFA WRITE? >>2D08
2CBB NO, GET HIGH BYTE OF BUFFER
2CF2 AND LOW BYTE OF BUFFER ADDRESS
2CF5 $42/43 >> FIRST PAGE OF BUFFER
2CF7 $4B/41 >> SECOND PAGE OF BUFFER >>2D23
2CF9 GET PAGE NUMBER (03C1)
2CFE $3C/3D >> BLOCK IN RAMDRIVE
2D00 $3B/3F >> SECOND PAGE OF S
2D06 ALWAYS BRANCH AROUND WRITE
ProDOS Relocator -- VI.2 -- 6 SEP 86

DESCRIPTION/CONTENTS

2D7D GOT A REMAINDER? 0
2D7F IF SO, DECREMENT <<2D81
2D81 THEN ADD INTO $00 (NOT USING 1)
2D82 TO FORM $10 THRU $11BIE
2D85 BLOCK*2 FOR PAGE NON ($3C1)
2D86 COPY THE BLOCK <CHUNK>
2D89 THEN EXIT >>$1DF ++

2D8C ************* READ/WRITE BITMAP BLOCK *************
2D8C USE RAMDRIVE BUFFER
2D91 USING BUFFER POINT (NO ACTUAL BITMAP BLOCK)
2D94 WRITING? >>2D9A, WRITES <<2E5
2D96 NO, READING = ZERO:
2D9B COPY BITMAP IMAGE TO RAMDRIVE BUFFER <<336
2DA3 COPY BLOCK BACK TO RAMDRIVE BUFFER <<3C2
2DA6 THEN EXIT >>3DE -- CALLER'S BUFFER <<3C5

2DA9 WRITING, COPY THIS:
2DAC SET UP BUFFER POINT TO RAMDRIVE BUFFER <<5C3
2DB1 COPY 16 BITMAP BYTES <<5E5
2DB3 INTO PAGE-3 BITMAPS FROM RAMDRIVE BUFFER
2DB9 THEN EXIT >>3DF -- IMAGE <<3C2

2DBC ************* RAM DRIVE DATA (AT $3BC) *************
2DBC FIRST TIME ENTRY F5
2DBD COMMAND FROM FARM AG
2DBE UNIT NUMBER FROM FARM LIST
2DBF BUFFER ADDRESS FROM FARM LIST
2DC1 BLOCK NUMBER FROM FARM LIST
2DC2 BIT MAP IMAGE FOR RAM DRIVE
2DD2 RAMDRIVE VOLUME NAME
2DD3 'RAM': <<3E5
2DD6 ACCESS, ENTRY LENGTH
2DD8 NUMBER OF ENTRIES IN FARM LIST
2DD9 FILE COUNT
2DBB BIT MAP BLOCK POINTER
2DBD BLOCKS ON DISK

2DDE ************ EXIT TO RAM IN MEMORY **************

2DDE WRITE ENABLE HIGH
2DE5 RESTORE BISTORE (BANK1) (C0UB)
2DE7 BISTORE WAS ON (PRESENT >>2DEA
2DEA GO AROUND MEMORY LIST
2DED LOW-ORDER BYTE AND SD BY XFER >>3EF
2DEE HIGH-ORDER BYTE NOW
2DFB RETURN TO >>F44 (CHECK BY XFER ROUTINE
NORMAL EXIT)
Supplement

2E82 **********
  V1.0 -- 6 SEP 86
  NEXT OBJECT ADDR: 2E7F
  2E82 SAVE $3ED8/ CONTENTS
  2E83

  FF84
  2E84 ZERO PAGE AUTO AREA

  2E90 **********
  (NOTE: $3EE
  FOR THI
  IS RESH)
  2E90 NOT USED SAVE AREA

  2F00 **********
  $2E90-$2EFF NOT USED
  THE AREA FROM $FF90-$FF99 IS RESERVED
  2F00 MLI LOAD (RAM CALLER. FROM $FF9B TO $FFFF
  ISERVED FOR THE IRQ HANDLER.)

START OF MLI LOAD IMAGE

IMAGE AT $2F00
2600 MODULE STARTING ADDRESS

**************************************************
* PRODOS RELOCATOR
* LOADED AS THE FIRST
* PORTION OF THE PRODOS
* IMAGE AT $2000.
* VERSION 1.3 -- 2 DEC 86
* **************************************************

THE 1.3 VERSION OF THE PRODOS RELOCATOR IS
INSTRUCTION FOR INSTRUCTION THE SAME FROM
$2000 TO $25F5 AND FROM $2800 TO $2EFF.
SOME ADDRESSES IN THESE AREAS CHANGE BECAUSE
THEY ARE ADDRESSES WITHIN THE MODIFIED PORTION
OF THE RELOCATOR OR THE MLI.

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

25B1 ----

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

25B1 CLICK SPEAKER (C83B)
25B4 STORE IN MAIN MEMORY (C83C)
25B7 80 COL DISPLAY OFF (C83D)
25BA SET NORMAL VIDEO <F84>>
25BD CALL MONITOR INITIALIZATION <FB2F>
25C0 SET VIDEO PR#0 <F93>
25C3 SET KEYS.setCode <F89>
25C6 OUT OF DECIMAL MODE
25C7 CLEAR SCREEN <FC5B>
25C8 PRINT "APPLE //" (2605)
25D7 PRINT "PRODOS 8 " ETC. ON ROW 12 (260D)
25E2 PRINT 12 BLANKS ON ROW 14 (262B)
25ED PRINT "COPYRIGHT" ETC. ON ROW 23 (2637)
25F9 PRINT "ALL RIGHTS RESERVED" ON ROW 24 (265E)
2681 CLICK SPEAKER AGAIN (C83B)
26B4 DONE

2685 ********** DATA

2685 'APPLE II' AREA ******************
269D 'PRODOS 8 V1'
2628 'COPYRIGHT 1983 2-DEC-86'
26E 'ALL RIGHTS '
265 'PLE COMPUTER, INC., 1983-86'
2672 8 BYTES FOR RESERVED.'
267A DRIVER ADDRES
267C SPACES LEFT SMARTPORT STATUS CALL
267D SLOT 2 FLAG (S)
267E ********** TRAIN = PRODOS STORAGE DEVICE IN SLOT 2)

2678 ----

2680 ZERO SOME THI
2687 DEVNT-$FF IN
268A ALL 14 DEVNINC
268F FIRST CHECK NO DEVICES YET) (BF31)
2693 IS A STORAGE:S ARE UNASSIGNED
2696 IF NOT, SET SLOT 2
2699 NOW POINT X80 DEVICE IN SLOT 2? <2BD>
269D STORAGE DEVN4 FLAG (267D)
26A0 NO: >>26F E\$LOT 7
26A2 GET &CSF BYTE IN SLOT? <2BD>
26A4 LOOK LIKE 16
26A6 LOOK LIKE 13
26A8 YES, DON'T US SECTOR DISK II. >>26B
26AA &CSF BYTE = DISK II STORAGE DEVICE ********
26AD CHECK BYTE A1
26AF TO IF LOW BYTE OF DEVICE ADDRESS (267A)
26B1 NOT A SMARTPORT OFFSET 7
26B3 GO DO SMARTPORTS A SMARTPORT
26B6 ---- UART INTERFACE >>26B6
26BD GET &CSFE (SORT STUFF >>26F3
26BA CAN WE AT LE.
26BE ANTICIPATE STATUS BYTE)
26BF CAN'T READ JUST READ STATUS AND DATA?
26C1 PUT LEFT UNION
26C5 PUSH CLC, INT. NO SENSE USING IT. >>26F
26C6 CARRY SET IMPOLE OF STATUS BYTE IN $12 <28C9>
26C7 GET HIGH BYVINDICATING ONE DRIVE
26C9 ALWAYS BRANw 2 OR 4 DRIVES

2685 ??? INTO DISK II PROCESSING >>26DB
ProDOS Relocator -- V1.3 -- 2 DEC 86

---

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
</table>
|      | **CONVERT STATUS FOR ID BYTE**  |}

---

**28C9**

- GET STATUS BYTE
- SHIFT LEFT NIBBLE TO RIGHT NIBBLE
- PUT IT IN $12
- RETURN

---

**28D4**

- CHECK FOR PRODOS STORAGE DEVICE
- RESET I/O CARD ROMS (CFFF)
- CHECK 3 BYTES ON CONTROLLER ROM
- ANTICIPATE FAILURE
- NOT A PRODOS STORAGE DEVICE >>28E6
- SUCCESS--THIS IS A PRODOS STORAGE DEVICE.
- RETURN

---

**28E7**

- COMMAND LIST FOR SMARTPORT CALL
- 3 PARAMETERS
- OVERALL STATUS CALL
- PUT STATUS DATA AT $2653
- STATUS CODE IS $00

---

**28EC**

- RELOCATION ROUTINE
- (X/Y REGS CONTAIN TABLE ADDR)
- SAVE PASSED TABLE ADDRESS
- ACCESS IIGS STATUS BYTE TO (C668)
- TURN OFF SLOT ROM, ENSURE ROM BANK 8
- ---
- GET OPERATION CODE
- VALID OPERATION? (4 OR LESS)
- NO, ERROR >> 2972
- $14/15 >> OUTPUT BLOCK
- $16/17 >> LENGTH
- NEGATIVE LENGTH? >> 2974
- CHECK OPERATION CODE
- ZERO BLOCK? >> 297D
- NO, $12/13 = $18/19 >> INPUT BLOCK
- $1A/1B >> END OF INPUT BLOCK
- COPY BLOCK ONLY? >> 29A1
- SAVE RELOCATION OPERATION CODE (2ACB)
- SAVE NUMBER OF RANGES TO CHECK (2ACC)
- ---
- COPY START PAGES TO TABLE
- ---
- AND FINALLY, RELOCATION FACTORS
- BUMP TO NEXT TABLE ENTRY <29A7>
ProDOS Relocator -- V1.3 -- 2 DEC 86
NEXT OBJECT ADDR: 2A3E

2A3F ************* ERROR RETURN ***********************
2A3F RETURN WITH POINTER
2A43 EXIT WITH ERROR CODE
2A46 RETURN

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

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

2A76 ************* COMPUTE INSTRUCTION LENGTH *******************
2A76 A-REG CONTAINS OPCODE
2A77 ISOLATE LAST 6 BITS AS TABLE INDEX
2A7C USE LAST 6 BITS AS TABLE INDEX
2A7E GET BYTE WITH 4 LENGTHS IN IT (2A8B)
2A81 ---
2A82 USING TOP TWO BITS AS INDEX... >>2A88
2A84 SHIFT DOWN THE PROPER LENGTH
2A86 AND ISOLATE IT IN A-REG
2A8A RETURN

2A8B ************* 6502 OP LENGTH TABLE ***********************
EACH BYTE CONTAINS FOUR 2-BIT LENGTHS
ProDOS Relocator -- V1.3 -- 2 DEC 86

NEXT OBJECT ADDR: 2A8B

ADDR DESCRIPTION/CONTENTS

2A8B ---

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

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

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

2AE5 NOT USED

THE REST OF THE RELOCATOR IS IDENTICAL TO VERSION 1.2
ProDOS MLI -- V1.2 -- 6 SEP 86
NEXT OBJECT ADDR: D700

D700 MODULE STARTING ADDRESS

*****************************************************************************
* * PRODOS MACHINE LANGUAGE INTERFACE * *
* THIS CODE IS MOVED INTO HIGH RAM ($DE00-$EFF) BY THE * *
* PRODOS RELOCATOR. * *
* IT PERFORMS ALL FILE MANAGEMENT * *
* AND OTHER SYSTEM FUNCTIONS AND * *
* SUPPORTS THE HARDWARE IN A * *
* DEVICE INDEPENDENT WAY. * *
* *
* VERSION 1.2 -- 6 SEP 86 * *
* *
*****************************************************************************

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

0040 Pointer to caller's parmlist
0041 -- device driver parmlist --
0042 Command
0043 Unit Number
0044 Buffer Pointer
0045
0046 Block Number
0047
0048 I/O Pointer - Index Block or...
0049 pointer into $8000 work buffer or...
004A caller's pathname buffer pointer
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 parmlist or...
004F old I/O buffer

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

0000 No Error
0001 Bad call type
0002 Bad parameter count
0025 Interrupt Table full
0027 I/O Error
0028 No device connected
002B Write protected

ProDOS MLI -- V1.2 -- 6 SEP 86
NEXT OBJECT ADDR: D700

002E Volume switched
0040 Invalid pathname syntax
0042 Too many files open
0043 Invalid REF NUM
0044 Nonexistent path
0045 Volume not mounted
0046 File not found
0047 Duplicate file name
0048 Disk full
0049 Volume Directory full
004A Incompatible ProDOS version
004B Unsupported storage type
004C End of file
004D Position past EOF
004E Access error
0050 File already open
0051 File count bad
0052 Not a ProDOS disk
0053 Bad parameter
0055 VCB overflow
0056 Bad buffer address
0057 Duplicate volume mounted
005A Bad volume bit map

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

0750 For direct movement of text to screen
07D0 07E1
07E2 07F0
07F1 07F7
07F2 07F8 Slot in use
07F3
07F4
07F5
07F6
07F7

D700 ******** RELOCATOR VARIABLE **********************

2278 Flag=1 when running on a 11GS

D700 ******** SYSTEM GLOBAL PAGE EQUIVATES **************

BF00 Jump to MLI entry point
BF03 JSPIARE (Jump to $E00F, QUIT code)
BF06 DATETIME vector
BF09 Jump to System Error
BF0C Jump to System Death Handler
BF0F System Error number
BF10 Device Driver address table
BF30 Slot/Drive last device
<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIP</th>
</tr>
</thead>
<tbody>
<tr>
<td>BF31</td>
<td>Count (-)</td>
</tr>
<tr>
<td>BF32</td>
<td>List of</td>
</tr>
<tr>
<td>BF58</td>
<td>Memory /</td>
</tr>
<tr>
<td>BF70</td>
<td>Open fi.</td>
</tr>
<tr>
<td>BF7E</td>
<td>Open fi.</td>
</tr>
<tr>
<td>BF90</td>
<td>Interrupt</td>
</tr>
<tr>
<td>BF9D</td>
<td>Interrupt</td>
</tr>
<tr>
<td>BF84</td>
<td>Interrupt</td>
</tr>
<tr>
<td>BF86</td>
<td>Interrupt</td>
</tr>
<tr>
<td>BF88</td>
<td>A reg s.</td>
</tr>
<tr>
<td>BF89</td>
<td>X reg s.</td>
</tr>
<tr>
<td>BF8A</td>
<td>Y reg s.</td>
</tr>
<tr>
<td>BF8B</td>
<td>S reg s.</td>
</tr>
<tr>
<td>BF8C</td>
<td>P reg s.</td>
</tr>
<tr>
<td>BF8E</td>
<td>Interrupt</td>
</tr>
<tr>
<td>BF90</td>
<td>Date/Tim</td>
</tr>
<tr>
<td>BF94</td>
<td>File op.</td>
</tr>
<tr>
<td>BF95</td>
<td>Backup</td>
</tr>
<tr>
<td>BF96</td>
<td>Temporal</td>
</tr>
<tr>
<td>BF9A</td>
<td>Prefix</td>
</tr>
<tr>
<td>BF9B</td>
<td>MLI act.</td>
</tr>
<tr>
<td>BF9C</td>
<td>Last MLI</td>
</tr>
<tr>
<td>BF9E</td>
<td>MLI X r.</td>
</tr>
<tr>
<td>BF9F</td>
<td>MLI Y r.</td>
</tr>
<tr>
<td>BFA0</td>
<td>HIGH RAM</td>
</tr>
<tr>
<td>BFDC</td>
<td>Interrupt</td>
</tr>
<tr>
<td>BF94</td>
<td>Bank sw</td>
</tr>
</tbody>
</table>

D700 ************

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIP</th>
</tr>
</thead>
<tbody>
<tr>
<td>C00C</td>
<td>Reset B.</td>
</tr>
<tr>
<td>C029</td>
<td>IIGS NE</td>
</tr>
<tr>
<td>C051</td>
<td>Set TEX</td>
</tr>
<tr>
<td>C053</td>
<td>Set Mix</td>
</tr>
<tr>
<td>C054</td>
<td>Display</td>
</tr>
<tr>
<td>C056</td>
<td>Set LOR</td>
</tr>
<tr>
<td>C083</td>
<td>Read/Write</td>
</tr>
<tr>
<td>C08B</td>
<td>Read/Write</td>
</tr>
<tr>
<td>CFFF</td>
<td>Reset a</td>
</tr>
</tbody>
</table>

D700 ************
<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>D918</td>
<td>not used</td>
</tr>
<tr>
<td>D919</td>
<td>not used</td>
</tr>
<tr>
<td>D91A</td>
<td>Bit Map Pointer</td>
</tr>
<tr>
<td></td>
<td>Block offset into multi-block bitmap of</td>
</tr>
<tr>
<td>D91C</td>
<td>next free bit.</td>
</tr>
<tr>
<td>D91E</td>
<td>Count of open files</td>
</tr>
<tr>
<td>D920</td>
<td>VCB1 through VCB7</td>
</tr>
<tr>
<td>DA00</td>
<td>*********** BITMAP BUFFER *********************************</td>
</tr>
<tr>
<td></td>
<td>Buffer 1st half</td>
</tr>
<tr>
<td>DB00</td>
<td>Buffer 2nd half</td>
</tr>
<tr>
<td>DC00</td>
<td>*********** PRIMARY BUFFER *******************************</td>
</tr>
<tr>
<td></td>
<td>(Used for several things. DIRECTORY block offsets are</td>
</tr>
<tr>
<td></td>
<td>mapped into it below)</td>
</tr>
<tr>
<td>DC04</td>
<td>Pointer Fields</td>
</tr>
<tr>
<td>DC05</td>
<td>** DIRECTORY HEADER ***</td>
</tr>
<tr>
<td>DC06</td>
<td>Type/Length (TTTTTTTTT)</td>
</tr>
<tr>
<td>DC14</td>
<td>Volume Name (Max 15)</td>
</tr>
<tr>
<td>DC1C</td>
<td>Reserved</td>
</tr>
<tr>
<td>DC1C</td>
<td>Creation Datetime</td>
</tr>
<tr>
<td>DC20</td>
<td>Version</td>
</tr>
<tr>
<td>DC21</td>
<td>Min Version</td>
</tr>
<tr>
<td>DC22</td>
<td>Access Byte</td>
</tr>
<tr>
<td>DC23</td>
<td>Entry Length</td>
</tr>
<tr>
<td>DC24</td>
<td>Entries per Block</td>
</tr>
<tr>
<td>DC25</td>
<td>File Count</td>
</tr>
<tr>
<td>DC27</td>
<td>Bitmap Pointer</td>
</tr>
<tr>
<td>DC29</td>
<td>Entry number within parent's block</td>
</tr>
<tr>
<td>DC29</td>
<td>Total Blocks</td>
</tr>
<tr>
<td>DC2A</td>
<td>Length of entries in parent</td>
</tr>
<tr>
<td>DC2B</td>
<td>(remainder of first page of block)</td>
</tr>
<tr>
<td>DD00</td>
<td>(second page of block)</td>
</tr>
<tr>
<td>DE00</td>
<td>*********** MLI MAIN ENTRY POINT **************************</td>
</tr>
<tr>
<td>DE01</td>
<td>Clear decimal mode</td>
</tr>
<tr>
<td>DE02</td>
<td>Retrieve status byte from stack</td>
</tr>
<tr>
<td>DE02</td>
<td>and store it in global page. (BF96)</td>
</tr>
<tr>
<td>DE05</td>
<td>Save Registers (BF9F)</td>
</tr>
<tr>
<td>DE0B</td>
<td>Set ($4B) -&gt; Address of function code -1</td>
</tr>
<tr>
<td>DE0F</td>
<td>Set CM subdivision -&gt; True return address</td>
</tr>
<tr>
<td>DE1C</td>
<td>Retrieve status byte, (BF96)</td>
</tr>
<tr>
<td>DE1F</td>
<td>push it onto the stack,</td>
</tr>
<tr>
<td>DE20</td>
<td>and pull it into status register.</td>
</tr>
<tr>
<td>DE24</td>
<td>Init Global Page System error to 0 (BF0F)</td>
</tr>
<tr>
<td>DE28</td>
<td>Get Function Code</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>DE2B</td>
<td>Build hash index into Command Table (X reg)</td>
</tr>
<tr>
<td>DE34</td>
<td>Is this function code valid?</td>
</tr>
<tr>
<td>DE39</td>
<td>No &gt;&gt;DEB1</td>
</tr>
<tr>
<td>DE3C</td>
<td>Set ($48) -&gt; Parameter list</td>
</tr>
<tr>
<td>DE49</td>
<td>Get parameter count required (FD4D)</td>
</tr>
<tr>
<td>DE4C</td>
<td>None? &gt;&gt;DE6A</td>
</tr>
<tr>
<td>DE4E</td>
<td>Is parameter count correct?</td>
</tr>
<tr>
<td>DE50</td>
<td>No &gt;&gt;DEB5</td>
</tr>
<tr>
<td>DE52</td>
<td>Check class of function (FD2D)</td>
</tr>
<tr>
<td>DE55</td>
<td>Quit?</td>
</tr>
<tr>
<td>DE57</td>
<td>yes &gt;&gt;DE67</td>
</tr>
<tr>
<td>DE59</td>
<td>no,</td>
</tr>
<tr>
<td>DE5A</td>
<td>$9X - Calls to I/O Drivers &gt;&gt;DE70</td>
</tr>
<tr>
<td>DE5C</td>
<td>$CX/$DX - Non System calls &gt;&gt;DE7B</td>
</tr>
<tr>
<td>DE5E</td>
<td>Else, $4X - Interrupt support</td>
</tr>
<tr>
<td>DE5F</td>
<td>Isolate type ($=ALLOC, 1=DEALLOC, 2=SPECIAL)</td>
</tr>
<tr>
<td>DE61</td>
<td>Call Interrupt Support &lt;DEFD&gt;</td>
</tr>
<tr>
<td>DE64</td>
<td>Then Exit to Caller &gt;&gt;DEB2</td>
</tr>
<tr>
<td>DE67</td>
<td>Go to quit code via global page &gt;&gt;BF03</td>
</tr>
<tr>
<td>DE6A</td>
<td>***************************************************</td>
</tr>
<tr>
<td></td>
<td>***** MLI GET TIME CALL *****</td>
</tr>
<tr>
<td></td>
<td>***************************************************</td>
</tr>
<tr>
<td>DE6A</td>
<td>Call Date/Time driver &lt;BF06&gt;</td>
</tr>
<tr>
<td>DE6D</td>
<td>and exit to caller &gt;&gt;DEB2</td>
</tr>
<tr>
<td>DE70</td>
<td>*******************************************************</td>
</tr>
<tr>
<td></td>
<td>***** MLI READ BLOCK CALL *****</td>
</tr>
<tr>
<td></td>
<td>***** MLI WRITE BLOCK CALL *****</td>
</tr>
<tr>
<td></td>
<td>*******************************************************</td>
</tr>
<tr>
<td></td>
<td>$80 - Read Block</td>
</tr>
<tr>
<td></td>
<td>$81 - Write Block</td>
</tr>
<tr>
<td>DE75</td>
<td>---</td>
</tr>
<tr>
<td>DE77</td>
<td>Set $42 -&gt; 1 for READ, 2 for WRITE</td>
</tr>
<tr>
<td>DE78</td>
<td>Do Block I/O &lt;DEBC&gt;</td>
</tr>
<tr>
<td>DE7B</td>
<td>Then Exit to Caller &gt;&gt;DEB2</td>
</tr>
<tr>
<td>DE7B</td>
<td>*******************************************************</td>
</tr>
<tr>
<td></td>
<td>$CX and $DX CALLS</td>
</tr>
<tr>
<td>DE7B</td>
<td>---</td>
</tr>
<tr>
<td>DE7C</td>
<td>Isolate function Index</td>
</tr>
<tr>
<td>DE7F</td>
<td>Perform function and exit to caller &lt;E03E&gt;</td>
</tr>
<tr>
<td>DEB2</td>
<td>*******************************************************</td>
</tr>
<tr>
<td></td>
<td>EXIT TO CALLER</td>
</tr>
</tbody>
</table>
Beneath Apple ProDOS Supplement

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

DEB2 Clear Backup
DEB8A Error occurred?
DEB8D Save test results
DEB9E Disable interrupts
DEB9F Roll out most recent "active" bit (BF9B)
DEB92 Get test results back
DEB93 Store in X reg
DEB94 Set up Return Address on stack (BF9D)
DEB9C Put test results on stack
DEB9E Put error code in A reg
DEB9F Restore X reg (BF9E)
DEA2 Restore Y reg (BF9F)
DEA5 Put error code on stack
DEA6 Get RAM/ROM orientation (BF44)
DEA9 Exit via RAM Global Page >>BF00

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

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

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

DEB1 ---
DEB3 Branch always taken >>DEB7

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

DEB5 ---
DEB7 Call System Error Handler <DEE1>
DEB8 Exit to Caller >>DEB2

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

DEBC ---
DEBE Save Old Processor Flags
DEBF Disable Interrupts
DECG Copy Parameters to $43-$47
DECH Save Starting Buffer Page in $4F
DECD Find last page + 1
DEDD Round up if Buffer not page aligned >>DED3
DED3 Is this Memory already in use? <<PC63>
DED6 Yes, then exit with error >>DEE0
DED8 No, do Block I/O <<DEE4>>
DEDE Error? >>DEE0
DEDD No, then exit normally
DEDF RETURN
DEE0 Error Exit
DEE1 Call System Error Handler <<BF09>>

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

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

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

DEFD ---
DEFF Install unclaimed interrupt handler?
DF01 No, normal ALLOC/DEALLOC >>DF09
DF03 Yes, install a handler for unclaimed interrupts. <<PD23>
DF05 Error? >>DF5
DF08 No, done.
DF09 Test bit 0
DF0A 1=DEALLOC >>DF38

DEFC ---
DF0C Look for empty slot (BF7E)
DF15 His Address better be non-zero
DF19 Store Address of His routine in Global Page (BF7E)
DF22 And return the position number we used
DF2B Exit
DF29 Skip this Vector
DF2B Last one?
DF2D No, check another >>DF0E
DF2F Yes, Table Pull Error
DF31 Always taken >>DF35
DF33 Bad Parameter Error
DF35 Call System Error Handler <<BF09>>

DEALOC

DF38 ---
DF3A Get Position Number
DF3C Can't be zero >>DF33
DF40 Or greater than 4 >>DF33
DF43 Make Index into Table from it
DF46 And zero His Vector (BF7E)
DF4D Then Exit
And stack (BPB6)

Is this enhanced ROM? (DFE1)

Yes, skip some stuff we used to have to do. (DFE6)

Reload X and Y (BPB6)

Disable I/O ROM (CFP)

Replace active slot number (C1B0)

Exit from interrupt. (DFE6)

**DFE1 ENHANCE FLAG.** Set to 1 by RELOCATOR if new type ROM found.

(That is, if ROM IRQ Vector jumps below $D000)

Unclaimed IRQ Count. Incremented when an interrupt is unclaimed (256 tries are allowed).

---

**DFE4 ********** IRQ Handler **********

---

DFE4

DF50 Save a reg from Monitor (BPB8)

DF51 And X, Y, S and P (BPB9)

DF5D Is this ROM enhanced? (DFE1)

DF60 Yes, skip three pulls. (DFE6)

DF67 And RTI Address (BPB6)

DF6E Replace stack to original condition (DFE7)

DF72 Save active slot index (DFE7)

DF75 In bottom half of stack?

DF76 Yes, pop off 16 bytes and save them

---

DF7A

DF81 Save $FA - $FE (top of zero page)

---

DF8B Is there a User Vector #1 (BPB1)

DF8E No >>DF95

DF90 Yes, call it <<DF93

DF93 His interrupt? >>DFBD

DF95 Is there a User Vector #2 (BPB3)

DF98 No >>DF9F

DF99 Yes, call it <<DF96

DF9D His interrupt? >>DFBD

DF9F Is there a User Vector #3 (BPB5)

DFAF No >>DFA9

DFA4 Yes, call it <<DF99

DFA7 His interrupt? >>DFBD

DFA9 Is there a User Vector #4 (BPB7)

DFAC No, didn't find service routine. >>DFB3

DFAE Yes, call it <<DF9C

DFB1 His interrupt? >>DFBD

DFB3 Allow 256 tries. (DFP2)

DFB8 then indicate error type 1 and call System Death Handler. (BPB0)

DFBD Interrupt Service

DFBF Restore zero page (BPB6)

---

**DFE44 Times 2.**

**DFE5 Store Command Number in OR **

**DFE4A And use it to index OR.**

**DFE4E Set up Jump Vector & address (BPB6)

**DFE51 .hander address (BPB6)

**DFE57 Signal Backup Required.**

**DFE5C PATHNAME not required.**

**DFE5E Get Index and times 2. (DFE7)

**DFE6A Next Object Addr: DFP3**

**DFE6D Date/Time in Lisc? (validity check <

---

**Beneath Apple ProDOS Supplement**

---

**ProDOS MLI -- V1.2 -- 8 SEP 86**

---

**ADDR DESCRIPTION/CONTENTS**

---

**DF4E ********** IRQ Handler **********

---

**ADDR DESCRIPTION/CONTENTS**

---

**DF50 Save a reg from Monitor (BPB8)

---

**DF53 And X, Y, S and P (BPB9)**

---

**DF5D Is this ROM enhanced? (DFE1)**

---

**DF60 Yes, skip three pulls. (DFE6)**

---

**DF67 And RTI Address (BPB6)**

---

**DF6E Replace stack to original condition (DFE7)**

---

**DF72 Save active slot index (DFE7)**

---

**DF75 In bottom half of stack?**

---

**DF76 Yes, pop off 16 bytes and save them**

---

**DF7A**

---

**DF81 Save $FA - $FE (top of zero page)**

---

**DF83**

---

**DF8B Is there a User Vector #1 (BPB1)**

---

**DF8E No >>DF95**

---

**DF90 Yes, call it <<DF93**

---

**DF93 His interrupt? >>DFBD**

---

**DF95 Is there a User Vector #2 (BPB3)**

---

**DF98 No >>DF9F**

---

**DF99 Yes, call it <<DF96**

---

**DF9D His interrupt? >>DFBD**

---

**DF9F Is there a User Vector #3 (BPB5)**

---

**DFAF No >>DFA9**

---

**DFA4 Yes, call it <<DF99**

---

**DFA7 His interrupt? >>DFBD**

---

**DFA9 Is there a User Vector #4 (BPB7)**

---

**DFAC No, didn't find service routine. >>DFB3**

---

**DFAE Yes, call it <<DF9C**

---

**DFB1 His interrupt? >>DFBD**

---

**DFB3 Allow 256 tries. (DFP2)**

---

**DFB8 then indicate error type 1 and call System Death Handler. (BPB0)**

---

**DFBD Interrupt Service**

---

**DFBF Restore zero page (BPB6)**

---

**E009 SYSTEM DEATH**

---

**E009 Save Error number in WORD (FE9D)**

---

**E00A Turn off BIOS caller"s beep code (FEB5)**

---

**E00D Select standard text**

---

**E010 Are we running on a**

---

**E013 No. >>EO1A HANDLER ******************001**

---

**E015 Yes, initialize IEGS**

---

**E017 by clearing NEWVTRK X-REG**

---

**E01F**

---

**E021 Blank next to last <DISPLAY: CO51**

---

**E024 print "INSERT SYSTEM DISK?" (CO28)**

---

**E027 on bottom row of ascii**

---

**E02D Get error number bac video**

---

**E02E Expect errors in RAM. (CO29)**

---

**E030 Make it ASCII**

---

**E038 Put error number on ow of screen and (**

---

**E03B Infinite loop >033 DISK AND RESTORE**

---

**E03E PERFORM FILE **

---

**E03E HOUSEKEEP delay to 0 to 97**

---

**E03E Save function index screen (07F)**

---

**E03E Save function index screen (07F)**

---

**E03E Save function index screen (07F)**
ProDOS MLI -- V1.2 -- 6 SEP 86

ADDRESS DESCRIPTION/CONTENT

E1F0 $(48) -> 2nd Block of Buffer (index)
E200 ----
E201 Search all Volume Control Blocks (D910)
E204 for the one which goes with requested unit (D801)
E209 ----
E20F Can't find matching Volume Control Block
E211 So die with error type $80 <BF0C>
E214 No Buffer in open File Control Block
E215 So die with error type $80 <BF0C>
E219 Is Volume mounted? (D900)
E21C No, keep looking >>E209
E21E Save Volume Control Block index (FE59)
E220 Exit normally
E223 ----
E225 This looks wrong!!! (FE5A)
E228 Bad Reference Number error
E22B RETURN

E22C ************************************************************************************

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

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

E22C Set $(4E) -> Data Buffer <F1F5>
E22F Set Length = 0
E239 Get Unit Number
E23B Do all Units? >>E244
E23D No, just one
E23F Set length = 16 (FEA2)
E242 Always taken >>E249
E244 If all Units
E246 Set Length = 256 (maximum) (FEA3)
E249 Is Buffer in main RAM? <PC46>
E24C No, then exit >>E281
E24E Yes, zero out Buffer
E253 ----
E258 Index into Data Buffer = $00 (FE62)
E25D Get Unit Number again
E25F Isolate valid bits
E261 Specific Unit requested? >>E282
E263 No, copy Device List from Global Page <E947>
E266 Save Device Count (FE85)
E269 Get last Device (FE92)
E26C Generate return data for it <E282>
E26F Bump data buffer index by 16 (FE62)
E278 Get next Device (FE85)
E27C And go do it >>E266
E27E When done, exit
E281 RETURN

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

ADDRESS DESCRIPTION/CONTENT

E165 Set Release to top of Path buffer (D700)
E166 Save Keyboya old Prefix if one exists) (D700)
E174 Copy Prefixly
E177 [preceded l
E17F Exit acquire type Error
E180 Bad File Ty
E182 ----
E183 RETURN

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

E1B4 **********

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

E1B4 **********

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

E1B4 **********
E282  Save Device Number  (BF30)
E285  Scan for the Volume Control Block  <E859>
E288  Error?  >>E2C5
E28A  No
E28E  Read block 2 (Volume Directory)  <EB9>
E291  Get Volume Control Block offset  (FE59)
E294  Volume Directory read OK  >>E2A5
E296  Bad read, save error number
E297  Any file open?  (D911)
E29A  Yes  >>E2A2
E29C  Zero out this VCB entry  (D900)
E2A2  Put error number in Accum
E2A3  Always taken  >>E2C5
E2A5  Volume name exist?  (D900)
E2A8  No  >>E2A9
E2AA  Yes, Files open?  (D911)
E2AD  Yes  >>E2B8
E2AF  No, set up Volume Control Block for new Vol  <EB84>
E2B2  Error?  >>E2C5
E2B4  No
E2B6  Was a duplicate Volume Control Block found?  (FE7D)
E2B9  Yes, then error  >>E2C5
E2BB  See if the same Volume is still there  (FE59)
E2C1  If not, Disk Switch Error
E2C3  Else, all is well - continue  >>E2B3
E2C5  **************** ERROR  *******************************
       Store code in data buffer entry
E2C5  ---
E2C6  Store Device Number in entry  <E2F8>
E2CB  Store error code next
E2CD  Duplicate Volume error?
E2CF  No - done  >>E281
E2D2  Store Device Number for duplicate next  (FE7E)
E2DA  No Duplicate now
E2B1  Exit with error
E2B2  RETURN
E2B3  ********** MAKE ONLINE VOLUME ENTRY  *******************
E2B3  Get name length for loop index  (D900)
E2BC  Copy name to Buffer entry  (D900)
E2F3  Done yet?  (FE80)
E2F6  No, do another  >>E2EC
E2F8  Yes, find current Buffer entry  (FE82)
E2FB  Store Device number  (BF30)
E303  Return to caller
E378 Error? Yes, checks

E37A Set BLKNUM ++
E37B Read block 1
E37C Entry number with exit >>E330
E37E None allocated Parent Directory block number (PE8E)
E37F Set (8) for any entry <<EE00
E38A Skip link point in the Parent Dir. block (FE10)
E38C ---- 1
E38D Count enties? Yes
E389 Skip to next (FE6)
E389 Save link 0
E38D Add 1 to blocks!
E38F and $200 to EVP

E392 in entry
E3A8 Loop until done used
E3AA Write back Blockmark (FD96)
E3AD Error? then exit:
E3B0 Start all over >>E39D
E3B2 ********** ZERO $60 >>E3BE
E3B3 Now that there's room >>E304
E3BE Return to caller 30

E3BF ********** LOAD >> Buffer

E3C0 Copy date/time <<FILE

E3C4 to my variables:
E3D0 Loop until done routine <<E3B2
E3D2 Did he give bank creation?
E3D3 Yes, carry on >>
E3D5 No, then call >>E3C4
E3D7 System Date Time (Creation)?
E3E8 If: Storage Type =E320
E3E2 force it as 0
E3E8 else use 05D, instead (BF90)
E3FA Find file name is $00, $01, $02 or $03
E3ED OR: Storage Type
E3FF Store Type/Link
E3F3 Isolate name <<(FE82)
E3F7 Copy file name to name length (D700)
E405 Copy caller's block (FE27)

NOTE: This length
E40D and copy <<File Entry Buffer (FE82)
E412 ---- Access Byte
E413 add AUX len & could be validity checked!!
E41C Copy Version and
E41F constants to
E428 Indicate a block
E42D Copy Directory to Min Version (0,0) (PDB8)

PDRY (Fe43)
used
Header Block number (FE22)

E43C Is this a Seedling file?
E43E Yes >>E475
E440 No, Directory file - Build Header
E442 Copy completed Directory entry ??
E445 to $FE00 buffer first (DC04)
E449 Loop until done >>E442
E44B Make Storage type SE in Header ?
E450 Put "NUTLON" (Author) in Reserved in $FE00
E458 and Version, Min Version, Access; E27
E45B Entry-length, File count and (CC2)
E45E Parent pointer from constants
E45F Loop until done >>E452
E463 EOF = $200 (FE3D)
E466 Copy Parent Block entry number (FD80)
E46D Loop until done >>E466
E46P Copy Parent entry Length (FE19)
E475 Allocate a new disk block <<E95C
E479 error? >>E481
E47A Store it in key pointer of entry E24
E480 and in BLKNUM for I/O
E484 Write zeroed (or DIR HDR) key old
E487 error? >>E481
E489 Bump parent's file count (FE18)
E491 Go update directory <<E929 (FE38)
E494 error? >>E481
E499 Checkpoint Volume Bit Map and exit <<E5B5

E499 ********** POINT $48/49 AT DIRECTOR
E499 $48/49 --> Entry
E49D Skip link pointers (+4) ...t >>EB76
E49F File entry number counter (FE26)
E4A2 ---- ENTRY ****************
E4A3 Skip to proper entry
E4A6 Add entry length (FE19)
E4AB (bump MB)
E4AF (store LS8)
E4B1 RETURN

E4B2 ********** UPDATE DIRECTORY(s) ****
E4B2 System date available? (BF90)
E4B5 no, forget it >>E4C2
E4B9 yes, copy to last modified data
E4C2 turn on SUBBIT (backup if appropriate)
E4CB set DEVNUM of parent (FE21)
E4D1 and BLKNUM (FE24)
E4D7 reread DIR block containing entry
E4DA error? >>EB71
E4DC Point to proper entry in buffer state (FE45)
E4E3 Copy constructed entry to buffer

<EB99>

(FE27)
E4EE Is this block the DIR HDR? ---
E4F9 no, write this modified dir block? ---
E4FC error? >>E4B1 directory block <EBD5>
E504 and then read DIR HDR block
E507 error? >>E4B1 ok <EBC9>
E509 in any case..
E50B copy back update file count
E514 and ACCESS byte (with Bacnt to HDR (FE1B)
E51A re-write the HDR block <Bkp> (FE18)
E51D error? >>E573
E51F is this the VOL DIR? (DCD7)
E526 yes, all done -- exit >>E54
E528 Is subdirectory, get PARBD1
E52B store in variable area (ENT ENTRY, (DC29)
E52E Get PARENT ENTRY_LENGTH, <$26>
E531 store in variable area (FDC2A)
E534 get parent block number (F19)
E53A read Parent Directory blockDC27)
E53D error? >>E573 back <EBC9>
E53F find entry for this subdi
E542 system date available? (directory <E499)
E545 no >>E554 if90)
E547 yes,
E54B copy system date/time to...
E54E modified date/time in enti...
E554 write it back <EBD5> error
E557 error? >>E573
E55B BLKNUM = HDR block number
E564 same block we have now? ==
E56B yes, go back and date stamp...
E56A no, temp >>E51F
E56E read HDR block <EBD9>
E571 and go back to date stamp...
E573 error? then exit >>Parent DIR >>E51F

E574 ********** NOT ProDOS VOLUME? ********** ERROR **************

E574 ---
E577 RETURN

E578 ********** IS THIS ProDOS VOLUME? ********** ERROR **************

E578 does previous block ptr ==
E586 no, not a ProDOS volume -- 0? (DCW8)
E588 else, (DC44)
E58D does VOL DIR's STORAGE ==
E58F no, error >>E574 ?E = $E or ?F
E591 else, ok
E592 RETURN

E593 ********** GET FILE ENTRY **************

E593 follow path to it's end <<E5A6>
E596 error? >>E5A5
E59B copy file entry
E5A3 and exit
E5A5 RETURN

E5A6 ********** FOLLOW PATH TO A FILE **************

E5A6 get base dir's data <<E723>
E5A9 error? >>E5FC
E5AB another subdirectory in the path? >>E5D4
E5AD no, at end of path
E5AF $48/$49 --> $F004 (HDR)
E5B7 copy part of HDR to file entry
E5C1 File type = $F (Directory) (FDB8)
E5C4 BLOCK = 2 (FE77)
E5C7 No. blocks used = 4
E5C8 EOF = $800
E5CC TYPE = subdirectory ($D0)
E5D1 return to caller
E5D3 RETURN

*** SCAN DIRECTORY FOR FILE ***

E5D4 indicate no free entry found as yet
E5D9 signal in HDR block
E5DA zero count of names examined
E5DF find name in block <B6CD>
E5E2 got it! >>E644
E5E4 not yet, how many entries expected? (FE60)
E5E7 less entry number I just searched >>E5F
E5EC more file entries left to search? >>E5FE
E5FA no, directory error
E5FC ---
E5FD RETURN

E5FE yes, update entries left counter (FE60)
E603 back to first buffer page ($49)
E605 check next block pointer (DC02)
E60D if zero, directory error >>E5FA
E612 read next block of directory >>EBC9
E615 no errors, loop back for more >>E5DA
E617 exit if error
E618 free entry found. ENTRIES ***
E61B yes >>E63B
E61D no, check parents (PE63)
E620 is there another? (DC02)
E625 no... >>E63B
E627 yes, free entry was
E630 first in that block will be... (PE24)
E635 indicate free entry
E638 find next index to available (PE63)
E63B exiting with error? in <E764>
E63C no more indices?
E63E else, path not for file. path, file not found >>E641
E640 RETURN ...

E641 file not found err
E643 RETURN ...

E644 advance to next...
E647 end -- save entry, dir in path <E75D>
E649 get type of entry no. and exit >>E685
E64F subdir?
E651 no, bad path then
E655 copy key block to >>E63B
E657 to BKNUM
E65A and to current DIR...
E664 go read key block... block no (PE22)
E667 error? >>E660...
E66C new file counting (PE10)
E675 check minimum var <E68>
E678 too new? >>E68B... (DC21)
E680 count bits in reg
E681 ... >>E684
E684 ---
E687 there must be 5
E689 (there are) >>E689
E69A or else, index bits on (normally $75)
E69B ---
E698 RETURN ...

E69F copy DIR HDR... E695
E692 and go scan for...

E695 ********** COPY DIRECTORY HDR **************************

E69B Copy...
E697 CREATION, VERSION, MIN_VERS, ACCESS, (DC1C)
E69A ENTRY_LEN, ENTRIES_PER_BLK, FILE_COUNT (FE12)
E6A0 volume directory? (DC04)
E6A7 if so, exit now >>E6B4
E6A8 else, copy PARENT_POINTER, (DC27)
E6AE PARENT_ENTRY_NO., and PARENT_Entry_LEN (FE80)
E6B4 RETURN

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

E6B5 compute entry number (FE1A)
E6BE save it (FE26)
E6C3 and the block it's in (PE24)
E6CC exit

E6CD ********** SEARCH ONE DIR BLOCK FOR FILE *********************

E6CD get entries in this block (FE1A)
E6D3 "$48/$49 --> first entry
E6D9 ---
E6DB skip HDR? >>E710
E6DD no, non empty entry?
E6E1 yes >>E6F0
E6E3 no, do we need one? (PE63)
E6E6 no >>E710
E6E8 yes, remember it <E6A5>
E6E9 don't need another one now (PE63)
E6EE skip to next entry >>E710
E6F0 get length of name
E6F2 count it (FE5F)
E6F5 save it for loop (FE80)
E6F8 same len as we are wanting? (D700)
E6FB no, skip it >>E710
E700 ---
E704 compare names (D700)
E708 we found it! exit
E70F RETURN

E710 skip to next entry (FE62)
E714 end of block? if so, exit >>E70F
E71A bump "$48/$49 by entry len
E721 and go check next >>E695
E723 ************ GET DIRECTORY DATA ************************************
E723 find base directory <E77C>
E726 error? >>E77B
E72C zero out my variables (FEB8)
E732 set up device number (BF30)
E738 copy DIR HDR to my variables <E695>
E741 copy TOTAL BLOCKS from VCB (D912)
E747 copy BIT MAP Pointer from VCB (D91A)
E74D copy Block No. of this directory (B046)
E753 make second copy of file count (F618)
E75D advance to next subdir in path <E764>
E760 and update index (FE82)
E763 RETURN

E764 ********** ADVANCE TO NEXT DIR NAME ******************************
E764 get this DIR's index (FE82)
E76F add len of name to move index to next name (FE82)
E776 still in prefix portion? >>E777
E771 no, now starting caller's path suffix (BF30)
E774 save last DEVNUM accessed (FE67)
E777 return with len of next dir in path (D7D0)
E77B RETURN

E77C ********** FIND BASE DIRECTORY *******************************
E77C ---
E77E get old PREFIXPTR (BF9A)
E780 fully qualified pathname? (FEB4)
E784 no >>E787
E786 yes, no old PREFIXPTR anymore
E787 save old prefix index (FEB3)
E78A DEVNUM=0 (BF30)
E78D ---

*** SCAN VCB'S FOR A MOUNTED VOLUME ***
E78F scan (D980)
E792 got one >>E79F
E799 else, bump to next VCB

*** FIND LAST DIR IN PREFIX OR TOL DIR ***
E79F store name length (FEB0)
E7A2 same name as in pathname? (D7D0)
E7A5 no -- skip it >>E794
E7B3 save VCB index (FEB9)
E7B6 DEVNUM = VCB's unit no. (D910)
E7BC BLOCK = 2 (read VOLDIR if no old PREFIX)

E7C4 get old prefix index (FEB3)
E7C7 ---
E7C8B accumulate a new index (FEB2)
E7CB no previous prefix? >>E7DD
E7CE find last name in prefix (D7B0)
E7D3 read prefix directory instead of vol dir (FEB8)
E7D9 read block <E8D9>
E7E0 error? >>E7E7
E7E2 is this the right directory? <E881>
E7E5 Yes--exit. >>E60B

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

E7E7 open files? (FEB5)
E7ED yes, give up now >>E80B
E7EF else, (FEB3)
E7F2 put back old prefix length (FEB2)
E7F5 copy DVCLST from global page <E876>
E7FB use last device accessed first >>E80C
E7FD if none, get last in my device table (BF31)
E806 volume not found error
E80B RETURN

E80C ---
E80F search for device in device table (FEB2)
E817 device not found >>E80B
E819 when found, make it active device (BF30)
E81E remove it from table (FEB9)
E821 find its VCB <E859>
E824 not found? >>E846
E826 volume mounted there? (FEB5)
E82C no >>E833
E82E yes, open files here? (D911)
E831 yes, skip it -- get next unit >>E7FD
E833 else read block 2 (vol dir)
E837 read volume directory <EBC9>
E83A error? >>E7FD
E83C mount volume on VCB <E8A7>
E83F error? >>E7FD
E841 is this his chosen volume? <E881>
E844 no, try again >>E7FD
E846 yes, exit

E847 ************** COPY GLBL DVCLST TO MY TABLE ***************
E847 start with last device (BF31)
E84A get a unit number (BF32)
E84F copy it to device table (FEB2)
E855 return count of devices (BF31)
E858 RETURN
E859 ********* SCAN VCB'S FOR

E859 ---
E85D scan VCB's for a given
E864 not it? >>E868
E866 is it, save VCB index
E869 and exit normally
E86A RETURN

E86B else, volume mounted
E86E yes >>E874
E871 no, save VCB index
E874 ---
E876 bump to next VCB
E878 and go look at it >>X
E87A not found...
E87B any free entries? if
E87D else, all is well
E87E VCB table full error
E880 RETURN

E881 ********* COMPARE DIR NAME

E881 ---
E88D check DIR type (DC04)
E889 VOL DIR or SUB DIR? not, error >>E87E
E88B neither >>E894
E88D yes
E88F store len of its name
E892 and go on >>E899
E894 error exit
E895 RETURN

E896 compare directory name
E89C no match? >>E894
E8A5 they match! exit
E8A6 RETURN

E8A7 ********* MOUNT NEW VOL

E8A7 volume mounted? (FE58)
E8AD no, continue >>E884
E8AF yes, same one as ones (DC04)
E8B2 if so exit, else fail

E8B5 ********* COMPARE VOL NAMES TO MAKE

E8B4 zero out VCB
E8BF is this a ProDOS volume? <E578>
E8C2 no -- exit >>E908
E8C4 duplicate vol in VCB's? <E930>
E8C7 yes -- exit with that one instead >>E90A
E8C9 get new volume's name length (DC04)
E8D0 add to VCB index (FE59)
E8D4 and copy to VCB name field in empty VCB (DC04)
E8DF store in VCB name len field (D900)
E8E2 copy DEVNUM to VCB unit field (BF30)
E8E3 copy total blocks to VCB (DC29)
E8F4 copy block no. of vol dir to VCB
E8FE copy bit map block no. to VCB (DC27)
E890A exit
E890B RETURN

E890C ********* COMPARE VOL NAMES TO MAKE

E890C Get length (DC04)
E891 Same in VCB? (D900)
E894 Save VCB offset (FE58)
E897 Different from VCB >>E924
E899 Store len to use as buffer index
E89A Add length to VCB offset to get (FE58)
E89D index into VCB (last char of name)
E8D1 Compare names (D900)
E824 SEC if no match
E82B CLC if match
E82C Restore VCB offset to X=REG (FE58)
E82F RETURN

E8930 ********* LOOK FOR DUPLICATE VOL

E8930 start with first VCB
E8932 ---
E8936 this VCB has same name? <E90C>
E8938 no >>E947
E893B yes, files open? (D911)
E893F no >>E92F
E8942 UNIT=0 (D910)
E8945 and exit with no error >>E94F
E8947 else,
E8949 bump to next VCB
E894D and loop >>E932
E894F exit no errors
E8950 RETURN
ProDOS MLI -- V1.2 -- 6 SEP 86  

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>E951</td>
<td>save flag (FE7D)</td>
</tr>
<tr>
<td>E954</td>
<td>and VCB index of duplicate vol (FE7E)</td>
</tr>
<tr>
<td>E957</td>
<td>exit with error</td>
</tr>
<tr>
<td>E958</td>
<td>RETURN</td>
</tr>
</tbody>
</table>

**E959**

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

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

E959 any free blocks counted in VCB? (FE59)
E962 yes >>E9B6

*** COMPUTE VCB FREE BLOCK COUNT ***

E964 no, how many bit map blocks are there? <EA88>
E967 save it (less 1) (FE64)
E96C zero scratch (will count free blocks) (FE4E)
E972 no block found yet
E977 checkpoint bit map buffer <EB76>
E97A error? >>E9CA
E97F BLKNUM = bit map pointer (D91A)
E989 read block buffer to <EB89>
E990 error? >>E9CA
E99E count free blocks marked <E9CB>
E991 drop no. remaining to do (FE64)
E994 none left? >>E99F
E996 some, BLKNUM = BLKNUM + 1
E99C go process that >>E989

E99F did we find a free bit? (FE59)
E9A3 no -- volume full >>E9C7
E9A7 save VCB bitmap block offset (D91C)
E9AA save free block count in VCB also (FE4F)
E9B6 are there enough to satisfy request? (D914)
E9C5 yes, exit
E9C6 RETURN

E9C7 volume full error
E9CA RETURN

E9CB ********** SCAN AND COUNT BITMAP BLOCKS ***********************

E9C8 scan through both buffer pages
E9D2 counting one bits <<E9F8>
E9DD ---
E9E0 found free block already? (FE63)
E9E3 if so -- done >>E9F7
E9E5 any blocks found yet? (FE4E)
E9E8 no >>E9F7
E9ED yes, compute total no. of bitmap blocks <EA88>
E9F1 less number remaining (FE64)

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

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>E9F4</td>
<td>gives bitflag</td>
</tr>
<tr>
<td>E9F7</td>
<td>exit map block with first free bit (FE63)</td>
</tr>
</tbody>
</table>

**E9F8**

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

E9F8 shift and:
E9F8 count bit in:
EA03 exit when that are on (FE4E)
EA07 RETURN byte goes to zero

EA08 ********** COUNT

EA08 no. bitmap blks - 1 **************

EA08 get blocks
EA14 ---- on vol count (-1) (FE59)
EA15 isolate that
EA16 for bit map nibble count of block count
EA19 RETURN up block count

EA1A ********** PRE

EA1A $A a block on disk  **************

EA1A save MSB ...
EA1D and LSB (FE64)
EA21 block num
EA24 volume size passed too big for (D913)
EA28 yes, errored? (FE64)
EA2B no, get bit >>EA9B
EA31 save it (1) position for block no.
EA35 divide bloc
EA38 giving block no. by 8 (FE64)
EA41 save byte offset as remainder
EA44 make quot/offset (FE6A)
EA47 remember went 2 into block index (FE64)
EA4A read bit which page in that block (FE66)
EA4D error? >>map block (after checkpoint) <EB43>
EA4F are we at EA97?
EA55 yes >> EA98 proper block of bitmap yet? (FE71)
EA57 no -- check EA98
EA5A errored? >>Wkpoint <EB76>
EA5C indicate EA97
EA65 DFNUM of block wanted in VCB (FE64)
EA68 read actual bitmap (FE6E)
EA6B error? >>R. block directly <EB87>
EA6D get byte EA97
EA78 which page offset into page (FE6A)
EA73 get bit pos (FE6C)
EA76 page 0 >>pattern to set (FE63)
EA78 no, turn EA80
EA7E end and continue on in page 1 (DB80)
EA80 turn bit over >>EA86
EA86 mark bitmap in page 0 (DA80)
EA8E count block needs checkpoint

Jk freed (FE8A)
**EB28 ******* GET NEXT Bitmap BLOCK ****************************

EB28 use blocks of vol to compute (FE59)
EB29 number of blocks in bitmap (D913)
EB30 just scanned last block? (D91C)
EB31 yes, no space >>EB72
EB32 no, get next block (D91C)
EB40 checkpoint old one <EB76>

** EB43 ********** READ

EB43 have we reached the end of the bitmap?
EB44 yes >>EB5C
EB45 no, check next block
EB51 error? >>EB56
EB56 get next bitmap - 6 SEP 86
EB5C was bitmap
EB5F yes >>EB66 CONTENTS
EB61 no, read it
EB64 error? >>EB56
EB66 save bitmap
EB69 (page number) BITMAP BLOCK ****************************

** EB70 exit

EB71 return did bitmap for this unit yet? (FE59)
EB72 disk full or bit map of some other unit <EB76>
EB75 RETURN

** EB76 ******* CHK modified? (FE6D)

EB76 --

EB77 needs check?
EB7A no >>EB71 block offset times 2 (FE59)
EB7C yes, write <> (D91C)
EB7F error? >>EB76
EB81 does it need?
EB86 exit

** EB87 ********** READ

EB87 save DEVNUM POINT VOLUME BITMAP ****************************

EB8A copy block
EB94 BITMAP BLOCK
EB9A set up read point? (FE6D)
EB9B *** READ it <EB91>

EB9A save byte C0 checkpoint now
EB9A device = 0
EB80 block = bi
EB9A point to bit! BITMAP ****************************
EB80 do the i/o
EB82 restore old (FE6E)
EB85 ok? >>EB8C offset wanted (FE59)
EB87 no, error
EB8C RETURN

command

:: OR WRITE BITMAP ***

::mand
::map device (FE6E)
::map block (FE6F)
::map buffer (EB92)
::EBBF>
::DEVNUM (BP30)
::exit

** EB9C ********** FIND A FREE Disk BLOCK AND ***

****** AND ALLOCATE IT

EB9C go read bitmap <EB43>
EB9F error? >>EAC4
EAA1 first page 0
EAA6 scan last page of bitmap for free block
EAAE bump tm page 1 of buffer (FE6C)
EABB bump page offset (FE6B)
EAB4 scan 2nd page too (DB00)
EABC bump page (FE6B)
EABF get next block <EB28>
EAC2 continue >>EAA1
EAC4 error exit

EAC5 save byte index (FE6A)
EAC8 shift combination of page no. and (FE4F)
EACB byte offset left 3 bits to make (FE4F)
EACE room for bit position.
EADD depending on buffer page ... (FE6C)
EAE2 reload bit pattern from page 0 ... (DB00)
EAED or page 1 (DA00)
EAFA shift bit pattern, bumping block no.
EAE8 until a one bit is found >>EAF0
EAF0 then shift it back the way it was
EAF1 (with that bit turned off) >>EAF3
EAF3 store LSB of block no. (FE4E)
EAF6 store updated byte-back in proper page

** NEXT OBJECT ADDR: EA96
**ProDOS MLI -- Vl.2 -- 6 SEP 86**

**ADDR DESCRIPTION/CONTENTS**

---

**EB9**

********** READ BLOCK DESIGNATED BY A,X **********

EB9 Put low byte of block number in BLKNUM
EB9 and high byte in BLKNUM+1
EBD0 Read a block <EBD9>
EBD1 RETURN

**EBD1**

********** WRITE BITMAP **********

EBD1 set up write command
EBD1 and go do it >>EBD4

**EBD5**

********** WRITE BLOCK **********

EBD5 set up write command
EBD5 and go do it >>EBD8

**EBD9**

********** READ BLOCK **********

EBD9 set up read command
EBD9 and go do it >>EBD0

**EBDB**

*********** READ OR WRITE BLOCK ***********

EBDB save I/O command
EBDB where is my buffer?
EBDF save flags
EBD9 disable
EBD9 Set low byte of Buffer pointer
EBD9 to zero
EBD9 Initialize Global Page System error to 0 (BP0F)
EBD9 set I/O transfer occurred flag
EBD9 set unit to do I/O on (BF30)
EBD9 if block I/O <EBD4>
EBD9 error? >>EBDC
EBD9 no errors, restore things and exit
EBD9 RETURN

**EBFC**

error exit
EBF2 RETURN

**EBFF**

*************** MLI GET MARK CALL ***************

EBFF copy mark to caller's list from FCB (F5A)
EC0F exit with no errors
EClO RETURN
EC9B sapling or tree are ok >>ED16

*** SEEDLING ***

EC9D seedling, check position (FE73)
EC9A if position is outside of block 0..
EC94 promote to sapling >>ED04
EC96 else, (DBBC)
EC9E go get key block (seedling data block) >>ED6F

*** NEED TO CHANGE DATA BLOCKS ***

ECB1 does old index block need dumping? (DB08)
ECB6 no >>ECBD
ECBB yes, do so <EE97>
ECBB error? >>ECEB
ECBD check storage type (FE5E)
ECB9 tree file?
ECBD yes >>ECEB
ECBE no, sapling (FE74)
EC99 is position in first index block?
ECCE no, need master index, subindex and data >>ED2F
ECEE yes, first index, reset flags <EE9F>
ECDF is this a seedling?
ECD2 if so, see if in first block >>EC9D

*** SAPLING ***

ECD4 no, sapling, read its only index block <EE2A>
EC7D error? >>ECEB
ECDC set block no. of index block
ECBD always branch >>ED16

ECEB Error exit

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

ECE9 reset flags <EE9F>
ECEC read master index block <EE2A>
EC9F error? >>ECEB
ECF1 make index into block from (FE74)
ECF4 MSB_of_position/2
ECFA is there a subindex there?
ECF8 yes! >>ED09
ED02 no, fall thru to make one

*** GET NEW INDEX BLOCK ***

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

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

ED16 make block no. out of position (FE74)
ED1F use as an index to examine index block
ED21 entry
ED27 if its zero...
ED28 need new data block
ED2F set flags for what to allocate (FE5A)
ED3B new index block being created?
ED3A zero data block in any case <ED57>
ED3D if not index block that's it >>ED79
ED3F Zero the Index Block I/O Buffer <ED45>
ED42 and continue >>ED79

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

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

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

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

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

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

ED79 --
ED80 save previous mark in my variables (D812)
ED86 set new mark in the PCB (FE72)
ED91 "$4A/$4B --> data block buffer"
ED93 "$4C/$4D --> start of the page in
ED95 the data block buffer which contains (FE73)
ED98 the mark.
ED9E exit

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

ED9F get flags (FE5A)
EDAF turn off low 3 bits (allocate no new
EDAE blocks to file) (D808)
EDAB RETURN

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

EDAB DIR file?
EDAD yes! >>EDB4
EDAF no, bad storage type error
EDB1 go to SYSERR <<BF09>
EDB4 else, get page distance (FE4E)
EDB7 make it into blocks (divide by 2)
EDBE new position beyond old? (FE73)
EDC1 yes >>EDD1
EDC3 else, use previous mark
EDC5 copy to BLKNUM <<EDDF>
EDC8 error? >>EDEE
EDCA count it (FE62)
EDCD more to skip? >>EDC3
EDCF no, got it >>ED79
EDD1 use next block pointer in DIR block
EDD3 copy to BLKNUM <<EDDF>
EDD6 error? >>EDEE
EDDB count it (FE62)
EDDD more to skip >>EDD1
EDDF got it now! >>ED79

*** COPY LINK TO BLKNUM ***

EDDF copy block number link
EDD1 to BLKNUM
EDE4 if non zero,
EDEA then go read block. >>EDF0
EDEC else, EOF error
EDEF ---
EDEF RETURN

---

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

EDF0 set block number to read
EDF4 store read I/O command
EDF9 read to $48/$49 buffer
EDFA read the block <EE50>
EDFB error? >>EE0C
EE02 copy block no. just read to PCB
EE0C exit

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

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

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

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

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

EE2A ---

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

EE2C save command
EE2F block no. is key block in PCB (FE5A)
EE34 use $48/$49 buffer

*** I/O BLOCK ***

EE36 set I/O command
EE38 and block no. (D80E)
EE42 must be non-zero block number
EE46 or horrible death!
EE48 fall through to read/write block (D801)

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

EE50 (xreg = buff ptr in zero page)
EE51 disable
EE52 set up buffer pointer
EE5D get DEVNUM from PCB (D801)
EE63 set I/O transfer has occurred flag
**CHECKPOINT DATA BLOCK BUFFER**

- Buffer pointer at $4A/$4B
- Point to block no. in FCB
- Go write buffer to disk
- Error? >>EEB4
- Go turn off $40 flag in FCB and exit >>EEAB

**CHECKPOINT INDEX BLOCK BUFFER**

- Checkpoint volume bitmap <EB76>
- Use $49/$49 buffer
- Block no. is current index block in FCB
- Set to write
- Go write it to disk <<EE36>
- Error? >>EEB4
- No longer needs checkpoint
- Set flags accordingly <<FS6A>
- Exit

**MLI OPEN CALL**

- Search path for file <F593>
- Found it? >>EEBE
- No, bad path error
- Exit >>EEC5
- Else, see if FCB already open on file <EF9B>
- For write, if not, continue >>EECB
- Else, file already open error
- Continue

---

**CHECKPOINT BITMAP & KEY BLOCK**

- Checkpoint bitmap buffer <EB76>
- Go write key block for file >>EE27
- Get access <<FS45>
- DIR file?
- No >>EF99
- Yes, we are only reading (ID)
- Update access flag in FCB (DB3)
- Write protected? >>EF15
- No, another FCB open on this file
- Yes, no touchie >>EEC3

- Storage type must be <FS4
- Or equal to <SD
- Else, storage type error >>EEC
- Copy key block, blocks used...
- EOF mark to FCB (FS5A)
- KEYNUM = key block number
- Store OBJNUM in FCB (FS62)
- Go check and assign I/O buffer error? >>EF65
- Find VCB and set buff ptrs
- Set current level in FCB (BF94)
- Seedling, sapling or tree? (DF)
- No, skip next stuff >>EF7C
- Yes, make current mark in FCB
- First index block to force a block
- Index blocks and BLOCK 0
- Zero mark wanted, however (PS)
- Go set mark to zero <EC32>
- OK? >>EF81
- No, save the error code (PE)
- Got and I/O buffer? (DB8B)
- No >>EF74
- Yes, free it <<FS6E
- Mark FCB not in use
- Exit with error
- Return

---

**Beneath Apple ProDOS Supplement**

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

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>EE68</td>
<td>Set unit no. from DEVNUM (BF30)</td>
</tr>
<tr>
<td>EE6D</td>
<td>No errors have occurred yet</td>
</tr>
<tr>
<td>EE72</td>
<td>Do block I/O &lt;&lt;DEE4</td>
</tr>
<tr>
<td>EE75</td>
<td>Error? &gt;&gt;EF7A</td>
</tr>
<tr>
<td>EE77</td>
<td>No, exit normally</td>
</tr>
<tr>
<td>EE79</td>
<td>Return</td>
</tr>
<tr>
<td>EE7A</td>
<td>Else, exit with error</td>
</tr>
<tr>
<td>EE7C</td>
<td>Return</td>
</tr>
</tbody>
</table>

**CHECKPOINT BITMAP & KEY BLOCK**

- Checkpoint bitmap buffer <EB76>
- Go write key block for file >>EE27

---

**Next Object Addr: EEC6**

---

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

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>EEC7</td>
<td>Error -- unsupported storage</td>
</tr>
<tr>
<td>EECB</td>
<td>Get PCB index (FS5A)</td>
</tr>
<tr>
<td>EED1</td>
<td>Free PCB found? &gt;&gt;EED7</td>
</tr>
<tr>
<td>EED3</td>
<td>No, all PCB's in use error</td>
</tr>
<tr>
<td>EED6</td>
<td>Return</td>
</tr>
<tr>
<td>EED7</td>
<td>Zero out unused PCB</td>
</tr>
<tr>
<td>EEE2</td>
<td>Copy file ID fields to PCB</td>
</tr>
<tr>
<td>EEE5</td>
<td>(DEVNUM, DIR HDR BLK, DIR BLK)</td>
</tr>
<tr>
<td>EEE8</td>
<td>DIR ENTRY NO.</td>
</tr>
<tr>
<td>EEEF</td>
<td>DIR entry type (FS22)</td>
</tr>
</tbody>
</table>
we read past EOF? >>F024

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

ADDRESS DESCRIPTION/CONTENTS

F001 LENGTH = EOF - current_mark (D815)
F019 are we already at EOF?' (FEA2)
F01C no >>F022
F01E yes, EOF error
F023 else, zero length request? (FEA2)
F029 no >>F02E
F02B yes, set mark and exit >>F0E1
F02E validity check data buffer <<F046>
F031 no good? >>F02B
F033 ok, get storage type for file <<F03D
F036 standard kind of file?
F038 yes >>F03D
F03A no, DIR file >>F1A3
F03D else, set mark (to read proper buffers) <<EC32
F040 error? >>F026
F042 set up buffer indexing <<F040
F045 move all that can be moved out of data buff <<F122
F04B newline or len=0? exit now! >>F02B
F04A newline enabled? continue block by block >>F03D
F04C at least 1 block's worth left to be read? (FE76)
F050 if not, never mind >>F01D
F052 if so, store block count wanted (FE77)
F055 get FCB flags <<F056
F058 data block modified?
F05A yes, continue block by block for now >>F03D

*** FAST DIRECT READ ROUTINE ***

F05C signal no read occurred yet (FE7A)
F05F read directly into caller's data buffer
F067 set mark/read data block to caller's buff <<EC32
F06A error? >>F05D
F06C bump buffer pointer to next location
F070 drop length remaining by 512 bytes (FE76)
F076 bump mark (FE73)
F07E and mark's HSB as necessary (FE74)
F081 check if we are out of index block (FE74)
F087 drop counter of multi-blocks (FE77)
F08A and keep on >>F099
F08C end of multi-block read, put ptra back <<F195
F098 request to read? (FE75)
F095 no, exit through finish-up >>F0E1
F097 yes, conventional block by block read then >>F03D
F099 crossed index block? go do set mark >>F067
F09B make index block offset from mark (FE74)
F0A4 BLKNUM = next block in index block
F0A8 zero entry?
F0B2 if so, no direct read can occur until next (FE7A)
F0B5 set-mark/read >>F0BA
F0B7 get MSB of BLKNUM
F0BA (put index ptr back)
F0BE finish setting BLKNUM MSB
F0C0 if no read occurred within setmark, (FE7A)
F0C3 go back to setmark call >>F067
F0C7 disable
F0CB do I/O to caller's buffer directly
F0CC do block I/O directly <DEE4>
F0CF error? >>F0D4
F0D2 go back for more >>F06C

*** ERROR CLEANUP ***

F0D4 ---
F0D5 ---
F0D6 set buffer ptrs/VCB <F195>
F0DA ---
F0DB finish up I/O <F0E1>
F0DF exit with error
F0E0 RETURN

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

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

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

F0F8 ---
F0FC back up pointer to data buffer by an
F0FE amount equal to the LSB of the mark (FE72)
F101 (which makes indexing easier)
F107 newline mode enabled? (D81F)
F10B no, CLC >>F117
F10D yes, SEC
F10E copy newline mask (FE79)
F111 and newline character (Dh8A)
F117 first char index is LSB of mark in YREG (FE72)
F11A $4C/$4D --> page containing mark
F11E request count LSB in XREG (FE75)
F121 exit

F122 ********** COPY FROM I/O BLOCK BUFF ****************************

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

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

F122 ---
F123 partial page to move? >>F12D
F125 no, any full pages left? (FE76)
F128 no, read complete >>F17C
F12A yes, drop MSB of request length (FE76)
F12D ---
F12E copy one byte $4C --> $4E
F132 check for newline if carry set >>F165
F134 ---
F135 end of requested chunk >>F158
F137 ---
F139 more bytes to copy >>F12E
F13B end of page, bump pointers
F13F bump new mark (FE73)
F147 finished first page of block buffer?
F14B if so, continue >>F12E
F14E no, need another block from disk >>F17F
F150 another page in request length? (FE76)
F153 no >>F16F
F156 more in this block-page? >>F15E
F158 no, on last page of block?
F15C no >>F161
F15E yes, drop request len by one page (FE76)
F161 back up to next byte again
F162 go copy next page >>F137
F165 check for newline
F16D not it, never mindl >>F134
F16F else, were we done with page?
F170 no >>F17C
F172 yes, bump pointer
F174 and mark (FE73)
F17C set overflow flag (read completed) (F194)
F17F update mark LSB (FE72)
F184 bump request count if necessary
F185 update count LSB (FE75)
F18B point beyond data in caller's buffer
F193 ---
F194 and exit
**ProDOS MLI -- V1.2 -- 6 SEP 86**

**ADDR** | **DESCRIPTION/CONTENTS**
--- | ---
F195 | ********** CLEANUP AFTER DIRECT I/O **********

F195 restore caller's data buffer pointer  
F1A0 go set buffers/find VCB and exit >>E1b2

F1A3 | ********** DIRECTORY FILE READ **********

F1A3 set mark/read <EC32>  
F1A6 error? >>F1D7  
F1A8 set up buffer indexing <F0F8>  
F1A8 move data from I/O buffer <F122>  
F1A8 need next block? >>F1A3  
F1B0 no, finish up I/O <F0E1>  
F1B3 ok? exit >>F1D5  
F1B5 not ok. EOF error?  
F1B8 no, out now >>F1D6  
F1BA yes, point beyond EOF anyway? <E079>  
F1BD zero out data block I/O buffer <ED57>  
F1C5 dummy up an empty DIR block with previous (D810)  
F1CB pointer and no forward pointer in I/O  
F1CA buffer.  
F1CC zero out current block no. (D810)  
F1D5 return to caller  
F1D6 RETURN

F1D7 | finish up and error exit >>F0DA

F1DA | ********** COPY CALLER'S I/O LENGTH **********

F1DA copy request length to LENGTH and  
F1DC a temporary variable  
F1ED pick up ACCESS flags for file (FE5A)  
F1F3 exit to caller  
F1F4 RETURN

F1F5 | ********** POINT $4E/$4F TO CALLER'S **********

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

F1F5 set up pointer  
F200 YREG --> PCB (FE5A)  
F201 AREG = storage type (D807)  
F206 exit

F207 | ********** COPY FILE MARK AND COMPUTE **********

****** AND COMPARE END MARK ******

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

**ADDR** | **DESCRIPTION/CONTENTS**
--- | ---
F207 | ---
F20D | copy file mark (D812)  
F213 | and set previous mark also (FE55)  
F216 | add length giving new mark in scratch area (FEA2)  
F21D | (3 byte addition)  
F225 | will new mark exceed EOF? (FE4E)  
F233 | return with carry set accordingly

F234 | ********** SET NEW MARK & EOF **********

F234 set up indexes <F266>  
F237 | set new EOF in PCB (FE52)  
F23D | and new mark (FE55)  
F243 | save new mark in scratch variable too (FE4E)  
F24A | does mark exceed EOF? <F266>  
F24D | if so, we must extend EOF <F225>  
F253 | save old EOF (D815)  
F25B | set new EOF to mark if necessary (FE4E)  
F261 | ---
F265 | exit

F266 | subroutine to set 3 byte indexes  
F26D | RETURN

F26E | ********** MLI WRITE CALL **********

****** ML1 WRITE CALL ******

F26E copy request length <F1DA>  
F272 | copy file mark <F207>  
F275 | extend EOF if needed <F250>  
F279 | write access enabled?  
F27B | yes >>F281  
F27D | no, access error  
F281 | check status of this device <F431>  
F284 | error? >>F2C1  
F286 | request length = ? (FEA2)  
F28C | no >>F291  
F28E | yes, exit through finish-up >>F0E1

F291 | find caller's data buffer <F1F5>  
F294 | check storage type  
F296 | if DIR file, error >>F27D  
F29B | set mark/read blocks <EC32>  
F29D | error? >>F2C1  
F29D | get PCB flags <F5D6>  
F2A0 | any new blocks needed?  
F2A2 | no >>F306  
F2A4 | yes, allocating them  
F2A6 | ---
F2A7 count number of blocks needed
F2AA store number needed (FE5C)
F2B0 see if the blocks are available <E959>
F2B3 no, disk full ->F2C1
F2B5 yes, get PCB flags <F5D6>
F2BB master index block needed?
F2BA no ->F2C9
F2BC yes, go add it <F381>
F2BF and go on if no errors ->F2D5

F2C1 error,
F2C2 set new mark/E0F <F234>
F2C6 and finish I/O, exit with error ->F0DA
F2C9 check PCB flags again <F5D6>
F2CC need sub-index block?
F2CE no ->F2D5
F2D0 yes, go do it <F3BD>
F2D3 error? ->F2C1
F2D5 buy a new block for data <F411>
F2D8 error? ->F2C1
F2DA get PCB flags <F5D6>
F2DD indicate index buffer changed
F2DF no new blocks needed now
F2E1 update PCB flags (D608)
F2EB make index block offset from mark
F2EF store new block no. in index block (FE4F)
F2EC and store it as current data block (FE5A)
F306 set up buffer indexing <F0FB>
F309 start writing <F311>
F30C go see if more blocks are needed ->F298
F30E I/O finish up when done ->F0E1

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

F311 ---
F314 lower request count by 1 (FE76)
F31C ---
F31D copy partial page from caller's data
F31F to I/O block buffer
F324 ---
F327 next page in caller's area
F32B bump mark by 2 (FE73)
F333 still in same I/O block page?
F337 yes ->F31C
F33A no, clear overflow (I/O incomplete) ->F361

F33C any complete pages left to write? (FE76)
F33F no ->F351
F341 yes, more in this page?
F342 yes ->F34A
F344 no, first block page?
F348 no ->F34D
F34A yes, one less complete page to do (FE76)
F34D readjust index
F34E continue with full page ->F324

F351 ---
F352 a few bytes left to write? ->F35E
F354 no, bump data buffer by 100
F356 and mark (FE73)
F35E set overflow (I/O complete) (F194)
F361 store LSB of mark (FE72)
F364 and of request count (FE75)
F368 indicate data block modified <F5D6>
F36B and DIR entry needs update
F371 advance pointer into caller's buffer (FE72)
F37C no, PCB flag to indicate write occurred <F2A2C>
F380 exit

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

(F381 add higher level <F3CA>
F384 error? ->F3C9
F395 going storage type <F200>
F389 tree?
F388 yes ->F392
F38D no, add another level <F3CA>
F390 error? ->F3C9
F392 buy another block <F411>
F395 error? ->F3C9
F397 make offset into current index block (FE74)
F399A from current mark
F39C point index to new block (FE4E)
F39B also save as current data block (FE5A)
F39E checkpoint bitmap & key block <EE7D>
F3998 error ->F3C9
F39A zero out new index block ->ED45

F38D ********* ADD NEW INDEX BLOCK *****************************

(F38D check storage type <F200>
F3C2 seedling? ->F3CA
F3CD no, read key index block <EE2A>
F3C7 and go add data block ->F392
F3C9 exit if error occurs
ProDOS MLI -- V1.2 -- 6

*** ADD A BLOCK ***

F3CA buy a block <F40>
F3CD error? >>F410
F3D2 save old key bl.
F3DO make new block
F3D7 and current ind
F3F0 store pointer to:
F3F3 in first position
F3FA checkpoint bitmap
F3FD error? >>F410
F3FF get storage typ
F404 upgrade it to a SEP 86
F407 indicate DIR ent
F410 exit

F411 *********** BUY A DIRT

F411 allocate a disk E<br INDEX LEVEL TO FILE
F414 error? >>F410
F416 get PCB flags (0)
F419 indicate DIR ent
F422 add 1 to block seek number (D88C)
F42F --- the key block (D88C)
F430 exit
F430 old block in PCB (D88F)
F431 old block
F431 *********** DO STATUS:
F431 of new index
F431 and new key block (E87C)
F431 get PCB flags (0)
F434 any buffers in (0)
F436 if so, assume ext higher type (D887)
F438 no, (D881) try needs update (D888)
F438 select new dev

*** STATUS CASK BLOCK ***************

F43E Save Unit Number block <EA9C>
F440 Save Block Number
F446 Indicate Status (5D6)
F44A Indicate Block try needs update
F44E Go do I/O <DEE4 in use for file
F451 Restore Block
F459 Exit

If NO I/O YET ***************

F25D6> use? (I/O activity)
its ok >>F42F
ca (BF38)

***

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

F45A check REF NUM >>F494
F45E specific close?

*** CLOSE ALL OPEN FILES ***

F460 no errors yet (FEB9)
F465 store PCB index (D8B1)
F469 get its level (F8B2) and<br LEVEL, skip it (BF94)
F46C if below system level, skip it (BF85)
F46F yes, skip it >>F485

55
F4C9  ****************************************************
F4C9  ***** MLI FLUSH CALL *****
F4C9  ****************************************************
F4C9  flush specific file?
F4CD  yes >> F4FA
F4CF  no, clear flush-all error code (FE89)
F4D2  do all FCBS
F4D4  set FCB index for next FCB (FE5D)
F4DB  is this file open? (D800)
F4DB  no >> F4E2
F4DD  yes, flush it << F4F2
F4E0  error? >> F4F7
F4E2  bump to next FCB (FE5D)
F4E8  and go flush it too >> F4D4
F4EA  ---
F4E8  return with error code if any (FE89)
F4F1  RETURN

F4F2  ************ FLUSH A FILE & UPDATE DIRECTORY **************
F4F2  find buffer/VCB <E1E2>
F4F5  no error? >> F504
F4F7  go handle close error >> F5C7
F4FA  zero out close-all error
F4FF  validity check REF NUM <B1C7>
F502  error? >> F4F7
F504  is write access allowed? (D809)
F509  no, exit >> F4E4A
F50B  has a write occurred since last flush? (D81C)
F50E  yes >> F517
F510  no, >> F5D6
F513  does anything need flushing anyway?
F515  no, then exit now >> F4E4A
F517  else, get FCB flags << F5D6
F51A  has data buffer changed?
F51C  no >> F523
F51E  yes, checkpoint it << E83
F521  error? >> F4F7
F523  get flags again << F5D6
F526  has index buffer changed?
F528  nm >> F52P
F52A  yes, checkpoint it << EE97
F52D  error? >> F4F7
F52F  ---
F536  copy file identifier data to my variables (D800)
F540  set DEVNUM (BF30)
F543  BLKNUM = current DIR block (FE25)
F549  read DIR block << EBC9>

F54C  ****************************************************
F54C  error? >> F4F7
F54E  copy directory header << E695>
F551  are we in block with this file's entry? (F2E7)
F55A  no >> F561
F55F  yes >> F568
F561  no, set new block number
F565  read it << EBD9
F56B  point at directory entry in block << E499
F56B  copy file entry from directory << E598
F571  copy blocks used count to entry (D818)
F57F  copy new EOF (D815)
F58A  and new key block no. (D80C)
F593  isolate new storage type (D805)
F59D  combine it with name length (FE2A)
F5A5  and update type/len field in entry (FE2A)
F5AB  write entry back to directory << E4B2
F5AB  error? >> F5C7
F5B0  turn off "write occurred" flag (D81C)
F5B8  same bitmap in memory (F2E4)
F5BE  no, exit now >> F5C5
F5C0  yes, checkpoint it also << EBD6
F5C5  no errors, exit
F5C6  RETURN

F5C7  ******************** CLOSE ERROR ********************
F5C7  is this a close or flush all?
F5CC  no >> F5D4
F5D0  yes, save error code (F809)
F5D3  RETURN
F5D4  else, real error right now
F5D5  RETURN

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

F5DD  ********* FILE ACCESS ERROR ********************
F5DD  exit with file access error code
F5E0  RETURN

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

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>F5E1</td>
<td>get storage type &lt;F200&gt;</td>
</tr>
<tr>
<td>F5E4</td>
<td>if DIR file...</td>
</tr>
<tr>
<td>F5E6</td>
<td>it's an access error &gt;&gt;F5DD</td>
</tr>
<tr>
<td>F5E8</td>
<td>else, save type for truncate to</td>
</tr>
<tr>
<td>F5E9</td>
<td>mess with.</td>
</tr>
<tr>
<td>F5E0</td>
<td>write access permitted? (D800)</td>
</tr>
<tr>
<td>F5E4</td>
<td>no, error &gt;&gt;F5DD</td>
</tr>
<tr>
<td>F5F6</td>
<td>check device status &lt;F431&gt;</td>
</tr>
<tr>
<td>F5F9</td>
<td>error? &gt;&gt;F5DD</td>
</tr>
<tr>
<td>F602</td>
<td>copy EO from FCB (D815)</td>
</tr>
<tr>
<td>F610</td>
<td>copy caller's new EO</td>
</tr>
<tr>
<td>F61B</td>
<td>compare old EO to new (F855)</td>
</tr>
<tr>
<td>F621</td>
<td>if less than or equal to... &gt;&gt;F628</td>
</tr>
<tr>
<td>F623</td>
<td>if greater... &gt;&gt;F63D</td>
</tr>
</tbody>
</table>

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

F628 new EO beyond old
F62F copy caller's EO to FCB
F63A exit by indicating flush needed >>FA2C

*** OLD EO > NEW EO ***
*** TRUNCATE FILE ***

F63D flush first <F4FA>
F640 error? >>F5E0
F642 $43/$49 --> end of data block I/O buffer
F64C compare current mark to new EO (F85D)
F659 it is prior to EO >>F762
F661 if past EO, force mark back to EO (F85D)
F672 construct EO block number and (F875)
F675 byte offset into block from new (F891)
F678 EO mark. (F876)
F690 on a block boundary? (F892)
F693 yes >>F6B2
F695 no, (F690)
F699 decrement block by 1
F6A7 but don't let it fall below 0
F6B2 copy key block number (F85D)
F6C1 set blocks freed to zero
F6C9 truncate file at new EO <FA3E>
F6CC save status
F6D4 set new key block in FCB (F8BA)
F6DA drop PCB block count by number (D818)
F6DD of blocks freed in truncate routine. (F8BD)
F6EA copy new storage type (F8BC)
F6F7 turn off all block allocation flags <ED9F>
F6FA update VCB free block count <F9BD>
F704 copy mark (D812)
F7BA **************************** ****** ****** MLI SET FILE INFO CALL ****** ****** ****************************
F7BD get the file entry <E593>
F7BE error? >>F7E4
F7BF indicate backup needed now, (BF95)
F7CE copy 13 chars from caller's list to, (FDD7)
F7D1 file entry staging area >>F7D8
F7D8 ---
F7DD if any spurious access bits are on...
F7E1 access error!
F7E4 RETURN
F7E5 else, anything in his modification date?
F7E6 no >>F7EE
F7EB yes, go update directory >>E4C2
F7EE no, use system date then update directory >>E4B2
F7F1 **************************** ****** ****** MLI RENAME CALL ****** ****** ****************************
F7F1 follow path to file <E5A6>
F7F4 ok? >>F833
F7F6 no, bad name?
F7F8 no, real error >>F812
*** RENAME VOLUME ***
F7FA yes, copy new name <<F917>
F7FD error? >>F812
F7FF get first length (D700)
F803 get next (D700)
F806 bad path if more than one name for vol >>F807
F80B files open on volume? (D911)
F80E no, continue >>F814
F810 yes, file open error
F812 ---
F813 RETURN
F814 make type/len for a VOL DIR HDR
F81B write new name to VOL HDR <<F903>
F81E error? >>F889
F825 copy new name to device's VCB (D700)
F831 exit, no errors
F832 RETURN
ProDOS MLI -- V1.2 -- 6 SEP 86

ADDR DESCRIPTION/CONTENTS

F8BF else, compatibility error
F8C3 copy new path again <F917>
F8C6 error? >>F809
F8CB get length of last name <FEB4>
F8D3 copy it and name to file entry buffer (D700)
F8E3 combine new len with type (D700)
F8EB DIR file?
F8EC no, go update entry and exit >>F905
F8ED yes, (FEB3)
F8F3 read key block of this subdirectory <EBC9>
F8F6 error? >>F809
F8FB copy new name to DIR HDR (D700)
F900 and update directory's key block <F908>
F903 error? >>F809
F905 go update directory entry and exit >>B4C2

F908 ********** COPY PATH TO BUFF & WRITE *******************************

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

F917 ********** POINT TO NEW NAME ***************************************

COPY TO BUFFER

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

F925 ********** LOAD PATH INDEX ********************************************

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

F932 ***************** MLI DESTROY CALL ******************************

F932 get file entry <E593>
F935 error? >>F97E
F937 find PCB if any <EF9B>
F94A PCB open? (F662)
F94D yes, file open error >>F97C
F94F no free blocks needed
F947 go compute VCB free block count <E959>
F94A ok? >>F950
F94C error, disk full?
F94E no, real error >>F97E
F950 DESTROY enabled in ACCESS? (F4E8)
F955 yes >>F95C
F957 no, access error

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

ADDR DESCRIPTION/CONTENTS

F95C check status of device (BF30)
F962 error? >>F97E
F964 point to key block (FE3B)
F973 DIR file?
F977 no >>F940
F979 yes, handle differently >>F9D8
F97C File open error
F97E ---
F97F RETURN

*** DESTROY NON-DIRECTORY FILE ***

F980 save the storage type (FE8C)
F987 set EOF to zero (FE8C)
F98D byte offset = $200
F992 "truncate" the file at EOF=0 <FA3E>
F995 if error >>F97E
F997 free the key block in volume bitmap (FE8B)
F9A0 error >>F97E
F9A2 mark the file as deleted in DIR
F9A7 decrement file count in DIR (FE1E)
F9B2 checkpoint volume bit map <EB7C>
F9B5 error >>F97E
F9B7 update free block count in VCB <F9B0>
F9B8 and go update the directory >>EB42

*** SUBROUTINE TO UPDATE FREE BLOCK ***

*** COUNT IN VCB ***

F9B0 add blocks freed to total free blocks (FE5C)
F9C0 in VCB. (FE8D)
F9D2 start next search for free blocks at
F9D4 start of bitmap. (D91C)
F9D7 exit

*** DESTROY DIRECTORY FILE ***

F9D8 DIR file?
F9DA no, error >>FA27
F9DC read volume bitmap block <EB43>
F9DF error? >>FA26
F9E1 BLKNUM = key block pointer (FE3B)
F9E8 read it <EBD9>
F9EE compute VCB >>FA26
F9F0 if DIR has any files... (DC25)
F9FA access error
F9FF write back block marking entry free (DC04)
FA05 error? >>FA26
FA07 if "next_pointer" is zero.... (DC02)
FA11 go back and pretend it's a seedling >>F997
ProDOS MLI -- V1.2 -- 6 SEP 86

**DESCRIPTION/CONTENTS**

**FB1A**
more index blocks (tree file)? (FE8F)

**FB1D**
yes, must be tree file >>FB85

**FB1F**
no, demote to seedling <<FB95

**FB22**
if error >>FB4E

***TRUNCATE SEEDLING FILE***

**FB24**
read key block <<FB4F

**FB27**
error? >>FB4E

**FB29**
EOP in first page? (FE92)

**FB2C**
 yes >>FB44

**FB2E**
EOP in second page?

**FB2F**
no, exactly 256 bytes >>FB4D

**FB31**
get byte offset (FE91)

**FB34**

**FB36**
zero bytes in second page (DD00)

**FB3C**
EOP in first page? (FE92)

**FB3F**
no, we're done. >>FB4A

**FB41**
yes, zero bytes in first page, too (FE91)

**FB4A**
them write block back and exit >>EBD5

**FB4D**
exit normally

**FB4E**
RETURN

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

**FB4F**
Put index block number in A,X (FE8A)

**FB55**
Go read the block >>EBC9

**FB5B**
********** DEVOTE FILE TO SMALLER FILE TYPE**********

**FB5B**
get high byte of index block (FE88)

**FB5B**
and low byte (FE8A)

**FB61**
free the index block in the volume bitmap <EA1A>

**FB64**
if error >>FB7C

**FB66**
Establish first block of old index block (DC08)

**FB69**
as new index block. (FE8A)

**FB73**
reduce storage type by one (FE8C)

**FB7B**
and exit

**FB7C**
RETURN

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

**FB7D**

**FB7F**
save BLKNUM

**FB85**
Save Y-register (index within block) (FE68)

**FB90**
if it is non-zero...

**FB97**
free the block in the volume bitmap <EA1A>

**FB99**
if error >>FB2AB

**FB9C**
Restore index to Y-reg (FE68)

**FB9F**
zero this entry

**FBA7**

**FBA8**
loop through all entries >>FB75

**FBA8**
save error message, >>FB85

**FBA8**
restore old BLKNUM

**FBB3**
and exit in any case >>FB85

**FB84**
********** ALLOCATE I/O **********

**FB84**

**FB86**
get I/O buffer page (FB76)

**FB89**
can't be below $800

**FB95**
else, error >>FBFF

**FB9D**
can't be above $B000

**FB9F**
else, error >>FBFF

**FBC4**
$4A/$5B --> I/O buffer

**FBC8**
must be page aligned

**FBC9**
check each page of I/O buffer <<FBF7

**FBD2**
prior allocation in <<FBFF

**FBDF**
if ok, mark each page of I/O buffer for <FC3A>

**FBE3**
in system memory bit system bit map (BP58)

**FBF0**
assign buffer number to buffer location allocated <FCJA>

**FBF8**
and save buffer location allocated BP58

**FBFF**
RETURN

**FBFF**
bad I/O buffer error

**FC02**
RETURN

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

**FC03**

**FC04**
AREG contains buffer <<FE8A

**FC07**
move buffer pointer to buffer <<FBFF

**FC10**
exit number *2 (BP68)

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

**FC11**
is buffer already free?

**FC16**
yes, exit >>FC36

**FC1A**
zero its address in <<FC03

**FC27**

**FC2B**
free each page in buffer <<FBFF

**FC2B**
by marking system bit map global page (BP6F)

**FC36**
exit

**FC39**
RETURN

**FC3A**
for <FC3A>

**FC39**
RETURN

**NEXT OBJECT ADDR: FB9F**
PC3A ********** LOCATE BIT MAP POSITION **********
(GIVEN PAGE NUMBER)
FC3A XREG contains page number
FC3B compute page number times 4
FC3C use as offset for bitmap (PDCB)
FC3D page number / 8 = byte offset
FC46 into bitmap
FC48 exit

PC49 ********** CHECK BUFFER VALIDITY **********
START > $200   END < $BF00
PC49 get buffer address (MSB)
PC4D must be >$200 else error >>FBFF
PC4F get length (FEA6)
PC55 compute last page no. of buffer
PC5A ---
PC61 may not extend into $BF00
PC63 else, error >>FBFF
*** CHECK IF BLOCK OF MEMORY IS FREE ***
PC66 ---
PC67 see if this page is allocated <FC3A>
PC6D if so, error >>FBFF
PC6F else, check other page also
PC73 then exit if both have been checked
PC74 RETURN

PC75 *********************** MLI GET BUFF CALL ***********************
PC75 get next available buffer
PC7A put its address in caller's parmlist
PC82 and exit
PC83 RETURN

PC84 *********************** MLI SET BUFF CALL ***********************
PC84 mark his buffer allocated
PC89 error? >>FCAB
PC8B get old buffer address (FEA9)
PC95 free old buffer's pages in map <PC28>
PC9C copy old buffer contents
PC9E to new buffer
FCAA then exit

Beneath Apple ProDOS Supplement

ProDOS MLI -- V1.2 -- 6 SEP 86
Beneath Apple ProDOS Supplement

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

FD10

FD10 restore original P-reg
FD11 if error number is zero, (BF0F)
FD22 then indicate no error; =>FD25
FD24 otherwise indicate error
FD25 RETURN

FD26 ********** INSTALL A SPECIAL IRQ HANDLER ********************
This routine calls a subroutine located
at $D400 in BANK2 of high RAM. It is called when
an MLI command $42 is executed. Its purpose
is to install a routine that handles unclaimed
interrupts. Apparently the user has to
provide the routine at $D400.

FD26 Switch to BANK2 of high RAM, (C000)
FD29 execute the program there, <D400>
FD2C then back to BANK1 (C00B)
FD2F and return.

FD30

FD30 DATA AREA

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

IN HASH CODE ORDER: IF COMMAND IS...

INDEX IS COMPUTED AS:
000D EFGH
+0000 ABCD

FD30 GET BUF
FD31 UNUSED
FD32 UNUSED
FD33 UNUSED
FD34 ALLOC INTERRUPT
FD35 DEALLOC INTERRUPT
FD37 UNUSED
FD38 READ BLOCK
FD39 WRITE BLOCK
FD3A GET TIME
FD3B EXIT
FD3C CREATE
FD3D DESTROY
FD3E RENAME
FD3F SET FILE INFO
FD40 GET FILE INFO
FD41 ON LINE
FD42 SET PREFIX
FD43 GET PREFIX
FD44 OPEN
FD45 NEWLINE

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

FD46

FD46 READ
FD47 WRITE
FD48 CLOSE
FD49 FLUSH
FD4A SET MARK
FD4B GET MARK
FD4C UNUSED
FD4D SET EOF
FD4E GET EOF
FD4F SET BUF

FD50

FD50 GET BUF
FD51 UNUSED
FD52 UNUSED
FD53 UNUSED
FD54 ALLOC INTERRUPT
FD55 DEALLOC INTERRUPT
FD57 UNUSED
FD58 READ BLOCK
FD59 WRITE BLOCK
FD5A GET TIME
FD5B EXIT
FD5C CREATE
FD5D DESTROY
FD5E RENAME
FD5F SET FILE INFO
FD60 GET FILE INFO
FD61 ON LINE
FD62 SET PREFIX
FD63 GET PREFIX
FD64 OPEN
FD65 NEWLINE
FD66 READ
FD67 WRITE
FD68 CLOSE
FD69 FLUSH
FD6A SET MARK
FD6B GET MARK
FD6C UNUSED
FD6D SET EOF
FD6E GET EOF
FD6F SET BUF

FD70

FD70 ********** MLI COMMAND ADDRESS TABLE ********************
ProDOS MLI — V0.2 — 6 SEP 86

Beneath Apple ProDOS Supplement

ProDOS MLI -- V0.2 -- 6 SEP 86
NEXr OBJECT ADDR: FD70

FD70 CREATE
FD72 DESTROY
FD74 RENAME
FD76 SET FILE INFO
FD78 GET FILE INFO
FD7A ON LINE
FD7C SET PREFIX
FD7E GET PREFIX
FD80 OPEN
FD82 NEWLINE
FD84 READ
FD86 WRITE
FD88 CLOSE
FD8A FLUSH
FD8C SET MARK
FD8E GET MARK
FD90 SET EOF
FD92 GET EOF
FD94 SET BUF
FD96 GET BUF

FD98 ********** MLI COMMAND INFO BYTE **********

PATHNAME FLAG
| REFERENCE NUMBER FLAG
| DATE/TIME STAMP FLAG
| COMMAND NUMBER

FD98 1 0 1 - 00
FD99 1 0 1 - 01
FD9A 1 0 1 - 02
FD9B 1 0 1 - 03
FD9C 1 0 0 - 04
FD9D 0 0 0 - 05
FD9E 0 0 0 - 06
FD9F 0 0 0 - 07
FDA0 1 0 0 - 08
FDA1 0 1 0 - 09
FDA2 0 1 0 - 0A
FDA3 1 0 0 - 0B
FDA4 0 0 1 - 0C
FDA5 0 1 0 - 0D
FDA6 0 1 0 - 0E
FDA7 0 1 0 - 0F
FDA8 0 1 0 - 10
FDA9 0 1 0 - 11
FDAA 0 1 0 - 12
FDAB 0 1 0 - 13

ProDOS MLI -- V0.2 -- 5
ADD? DESCRIPTION/CONTENTS

SEP 86
N埃尔 OBJECT ADDR: FDAB

FDAC ********** C0NSTANTS **********

FDAC Blocks Used
FDAC End of File
FDAC Special ID (Misc)
FDAC 'HUSTON!' (it be 5 bits on)
FDAC Previous Block / Author's name
FDAC The following
FDAC Version of FC00 IS COPIED TO SUBDIR HDR + $20
FDAC Minimum Version Of FC00
FDAC Access Byte
FDAC Entry Length = RN|8|000|W|R
FDAC Entries per Block
FDAC File Count Block
FDAC Parent 189 (00)
FDAC Applied to SUBDIR HDR +$20
FDAC File Type (DAT)
FDAC Block Number (Factory)
FDAC Number of Blocks (Factory)
FDAC End of File

FDDB ********** BITS -- DATA AREA **********

FDDB 10000000
FDDB 01000000
FDDB 00000000
FDDB 00010000
FDDB 00001000
FDDB 00000100
FDDB 00000010
FDDB 00000001

FD3 ********** OFFSETS

FD3 (#CB's are at $ INTO FILE CONTROL BLOCKS **********

FD3 Key Block
FD5 # Blocks Used
FD7 End of File

FDAA ********** SETS — ERROR

FDAA Access
FD0 File Type
FD0C Aux Type
FD0E Storage Type
FD0F Blocks Used (NS

NS on means GET only no SET
FDE1 Datetime (Last Mod)
FDE5 Datetime (Creation)
FDE9 *********** FATAL ERROR MESSAGE ****************************
FDE9 ' INSERT SYSTEM DISK AND RESTART - ERR 0 '
FE11 ---
FE11 ********** VARIABLES - DATA AREA ***********************
FE11 Parent Pointer Block
FE13 Parent Entry Number
FE14 Parent Entry Length
FE15 Datetime (Creation)
FE19 Version
FE1A Min Version
FE1B Access Byte
FE1C Entry Length
FE1D Entries per Block
FE1E File Count
FE20 Bit Map Pointer
FE22 Total Blocks
    THE FOLLOWING 6 BYTES UNIQUELY IDENTIFY A FILE:
FE24 Device Number
FE25 Current Directory Block Number (HDR)
FE27 Block Number of File Entry in Directory
FE29 File Entry Number in Directory
FE2A ********* FILE ENTRY BUFFER ******************
FE2A Type/Length (TTTTLLLL)
FE2B File Name (Max 15) >>000F
FE3A File Type
FE3B Key Pointer
FE3D Blocks Used
FE3F End of File
FE42 Datetime (Creation)
FE46 Version
FE47 Min Version
FE48 Access Attribute
FE49 Aux Type (Load Address/Record Length)
FE4B Datetime (Last Mod)

FE4F Header Pointer
FE51 *********** Variable Work Area ***********************
FE51 3 Byte Scratch
FE54 ---
FE55 End of File
FE58 Previous Mark
FE5B Compare Vol Name Scratch
FE5C Offset into VCB Table ($D900)
FE5D Offset into FCB Table ($D800)
FE5E Free FCB found Flag
FE5F Number of Free Blocks needed
FE61 Storage Type
    Number of Entries Examined or...
FE62 FCB already open flag
FE63 File Count
FE65 Entries/Block Loop Count/Free FCB's refnum
    Free Entry Found Flag (if > 0) or...
    # of 1st bitmap block with free bit on or...
FE66 bit for free
FE67 $ Blocks in Bitmap left to search
FE68 Y Register temp
FE69 Pathname Length
FE6A Devnum for Prefix Directory Header
FE6B Block of Prefix Directory Header
FE6D Bitmap Byte Offset in Page
FE6E Bitmap Page Offset
FE6F Bitmap Buffer Page (0 or 1)
FE70 Bitmap Flag (if $00, needs writing)
FE71 Bitmap DEVNUM
FE72 Bitmap Block Number
FE74 Bitmap Block offset for Multiblock Bitmaps
    New Mark to be Positioned to for Set Mark
    or New Moving Mark (for READ)
FE75 or New EOF for SET_EOF
FE78 Request Count (Read/Write etc.)
FE7A Multi-Block I/O count
FE7B Newline character
FE7C Newline mask
FE7D I/O Transfer occurred flag
<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>FE7E</td>
<td>MLI Command * 2</td>
</tr>
<tr>
<td>FE7F</td>
<td>ORed into Access Flags ($20 - Backup)</td>
</tr>
<tr>
<td>FE80</td>
<td>Duplicate Volume Flag (if $FF)</td>
</tr>
<tr>
<td>FE81</td>
<td>Duplicate Volume's VCB index</td>
</tr>
<tr>
<td>FE82</td>
<td>MLI function code (low 5 bits)</td>
</tr>
<tr>
<td></td>
<td>Characters in current Pathname indx lvl or</td>
</tr>
<tr>
<td>FE83</td>
<td>ONLINE: volname len - loop index</td>
</tr>
<tr>
<td>FE84</td>
<td>new pathname: index to last name</td>
</tr>
<tr>
<td></td>
<td>old pathname: index to last name or..</td>
</tr>
<tr>
<td>FE85</td>
<td>ONLINE: index to data buffer</td>
</tr>
<tr>
<td>FE86</td>
<td>Old PFIXPTR value</td>
</tr>
<tr>
<td>FE87</td>
<td>Pathname fully qualified flag (if $FF)</td>
</tr>
<tr>
<td></td>
<td>Pathname: temp save area for index or..</td>
</tr>
<tr>
<td>FE88</td>
<td>ONLINE: DEVCNT</td>
</tr>
<tr>
<td>FE89</td>
<td>close-all error code</td>
</tr>
<tr>
<td>FE8A</td>
<td>Set EOF: new Key Block pointer</td>
</tr>
<tr>
<td>FE8B</td>
<td>New storage type (SET_EOF)</td>
</tr>
<tr>
<td>FE8C</td>
<td>Freed Blocks count</td>
</tr>
<tr>
<td>FE8D</td>
<td>EOF Block number (MSB then LSB)</td>
</tr>
<tr>
<td>FE8E</td>
<td>EOF byte offset into Block</td>
</tr>
<tr>
<td>FE8F</td>
<td>EOF - Master index counter</td>
</tr>
<tr>
<td>FE90</td>
<td>Save area for index into table below</td>
</tr>
</tbody>
</table>

**FE95 ********** DEVICE TABLE BUILT BY ONLINE **********************

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

| FE95 | device table part one |
| FE96 | device table part two |

| FE97 | length of path, etc. |
| FE98 | next buffer address |
| FE99 | 16 byte stack save area |
| FE9A | 6 byte zero page save area |
| FE9B | Jump Vector, used for indirect jumps |

**FE9C ********** $PEBF-$PEFP NOT USED ***********************

| FE9C | not used |
**** DESTROY DIRECTORY FILE ****

F9DE   DIR file:
F9E6   no, error >>FA2D
F9E2   read volume bitmap block <EB43>
F9E5   error? >>FA2C
F9E7   BLKNUM = key block pointer (FE64)
F9F1   read it <EB09>
F9F4   error? >>FA2C
F9F6   if DIR has any files... (DC25)
F9A0   access error
F9A5   write back block marking entry free (DC04)
F9A9   error? >>FA2C
F9AD   if "next_pointer" is zero... (DC02)
F9A7   go back and pretend it's a seeling >>F99D
F9A9   else, (DC03)
F9AC   free next block <EA1A>
F9AF   error? >>FA2C
F9A1   BLKNUM = next block (DC02)
F9A7   read it <EB09>
F9A2   if ok, continue in loop >>F9AD
F9A2   else, error exit
F9AD   incompatible file format error
FA32   ********* SET WRITE OCCURRED FLAG ***********
FA32   save some registers
FA35   indicate write occurred (FE86)
FA40   restore registers and exit
FA43   RETURN
FA44   ********** TRUNCATE FILE AT EOF **********
FA44   check storage type*16 (FE85)
FA47   seeling?
FA49   yes >>FA5B
FA4B   no, sapling?
FA4D   yes >>FA5B
FA4F   no, tree?
FA51   yes >>FA5B
FA53   General error--wrong storage type
FA55   jump to system death <BF0C>
FA58   go to seeling truncate >>FB2F

PREVIOUS CONTENTS

STARTING ADDRESS

***** PRODOS MACHINE LANGUAGE INTERFACE *****

PRODOS MLI *****

ADDR D700  MLI

1.3 VERSION OF THE PRODOS 8 MLI IS THE SAME AS VERSION 1.2, INSTRUCTION FOR INSTRUCTION,
FROM THE START ($DE80) TO ADDRESS $F992. SOME
VALUES BEFORE $F992 CHANGE BECAUSE THEY REFER TO
ADDRESSES GREATER THAN $F992.

ONLY THE PART OF THE MLI FROM $F980 TO $FFE7 IS
DOCUMENTED HERE FOR VERSION 1.3. REFER TO THE
1.2 VERSION FOR THE FIRST PART OF THE MLI.

***** DESTROY NON-DIRECTORY FILE *****

The storage type (FE85)
EBE to zero (FE85)
offset = $200
on destroy flag (FE88)
update the file at EOF=0 <FA44>
off the destroy flag (FE88)
off during truncation >>F97E
se the key block in volume bitmap <EA1A>

F980   >>F97E
the file as deleted in DIR
increment file count in DIR (FE47)
increment volume bitmap <EB76>

F98E   >>F97E
set free block count in VCB <F9C3>
F9D8   set up the directory >>EB82
F999   ** SUBROUTINE TO UPDATE FREE BLOCK ***
F998   ** COUNT IN VOLUME CONTROL BLOCK ***
F99B   ** Blocks freed to total free blocks (FE85)
F996   <<<< VCB. (FE86)
F99A   <<<< next search for free blocks at
F9AD   <<<< of bitmap. (D91C)
F9BB   <<<<
F9BD   <<<<
F9CB   <<<<

F9C3   <<<<
F9C6   <<<<
F9DB   <<<<
F9DA   <<<<
F9DD   <<<<

PRODOS Supplement
FA5B go to sapling truncate tree.
FA60 at most 128 blocks.
FA63 read the master index.
FA66 error? >> PAC8.
FA68 at EOF yet? (FEBC)
FA6E yes >> PAC9

*** FREE WHOLE node >> PAC6
(free 8 subindex)
master index block
share its buffer; a master
METER EOF ***
FA70 copy up to 8 non-zero
we must
FA75 a handy table (FEBC)
FA8E << key
FA93 if there weren't 8 blocks!
FA96 remainder of the block.
FA98 read the sub-index.
FA9F if error, zero all its blocks.
FAA7 >> FAC9
FAA9 (exit when a block)
FAAB read the sub-index.
FAAF quit if error.
FAB5 zero all its block.
FAB8 or swap pages (if (FC6C) D9)
FABC quit if error.
FABA write the former counter (FEB0) << BB95.

FA C3 and loop until all (FEB0) >> B95.
FA C5 then go back and read
FA C6 normal exit
FA C8 RETURN

FA C9 now go free all the nodes (if true)
FA CD which (if EOF=1) destroy mode.
FA DE if error >> PAC8 >> PAC9
FA D2 write back master index block (FBEB).
FA D5 if error >> PAC9
FA D7 EOF in first sub-index.
FA D8 if so, demote to master.
FA DC else, BLKNUM = sub-index.
FA DF contains the EOF
FA E4 (exit if none there.
FA E6 else, read first page (DC08)
FA EE and treat it as a
FA F0 unless there is an

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

*** TRUNCATE SAPPLEG FILE ***

FAF6 read index block (FB5A)
FAF9 if error >> PAC8
FAF8 index of last block in the file (FB59)
FAFE add one to point past end of file.
FAF9 if zero, no blocks to free >> FB0B
FB01 zero blocks past EOF (when truncating) (FB57)
FB04 or swap bytes for all but first block (when destroying).
FB07 if error >> PAC8
FB06 write back modified index block (EBD5)
FB09 if error >> PAC8
FB0B index of last block in file (FB59)
FB0E this index block is empty! >> FB25
FB07 Get BLKNUM of last data block (DC08)
FB0E (no block allocated?) >> PAC7
FB17 read in last data block (FB59)
FB22 and treat it as a seeding file >> FB34
FB24 unless error occurred.
FB25 more index blocks (tree file)? (FB58)
FB28 yes, must be tree file >> FB0D
FB2A no, demote to seeding (FB63)
FB2D if error >> PAC9

*** TRUNCATE SEEDLING FILE ***

FB2F read key block (FB5A)
FB32 error >> FB59
FB34 EOF in first page? (FB5B)
FB37 yes >> FB3F
FB39 EOF in second page? (FB5A)
FB3A no, exactly 256 bytes >> FB58
FB3C get byte offset (FEBA)
FB3F ----
FB41 zero bytes in second page (DD08)
FB47 EOF in first page? (FB5B)
FB4A no, we're done. >> FB55
FB4C yes, zero bytes in first page, too (FB58)
FB55 then write block back and exit >> EBD5
FB58 exit normally
FB59 RETURN
FB5A **************************** READ INDEX BLOCK ****************************

FB5A  Put index block number in A,X (FEB3)
FB60  Go read the block >>FB99

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

FB63  get high byte of index block (FEB4)
FB66  save it on stack
FB68  get low byte (FEB3)
FB6B  save it, too
FB6C  free the index block in the volume bitmap <E1A>
FB6F  restore the block number of the index block
FB70  set to zero page.
FB75  if error writing bitmap >>FB94
FB77  New index block is first block (DC00)
FB7A  from old index block. (FEB3)
FB83  For first entry in old index block,
FB85  zero the block number (if truncating) <FB7C>
     or swap the bytes (if destroying).
FB89  reduce storage type by one (FEB5)
FB91  Write the deleted index block back out. <EBD5>
FB94  RETURN

FB95  **************************** FREE ALL BLOCK NUMBERS IN ****************************
     *** AN INDEX BLOCK ***

FB95  ---

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

FB97  save BLKNUM
FB9D  Save Y-register (index within block) (FE91)
FBAD  if it is non-zero....
FBAF  free the block in the volume bitmap <E1A>
FBB2  if error >>FB6B
FBB4  Restore index to Y-reg (FEB9)
FBB7  zero this entry (when truncating) or <FB7C>
     swap the two bytes when destroying).
FBB8  ---
FBBB  loop through all entries >>FB9D
FBBE  save error message, if any
FBCC  restore old BLKNUM
FBCE  and exit

FBC7  **************************** ZERO OR SWAP INDEX BLOCK BYTES ****************************

FBC7  --- >>0001
FBC7  This a destroy operation? (FEB8)
FBCA  Yes, swap the two bytes. >>FBCF
FBCD  No, zero both bytes.
FBCD  A 65C02 instruction:1 >>FB05
     Note: we think this 65C02 instruction (BRA)
     snatch in by accident. It is the only
     65C02 instruction in all of ProDOS and
     means the 1.3 version won't run on Apple II
     Is that have a 65C02. But it is easy to
     patch this just change the byte at $FB0D
     ($4CD in the load image) from $80 to $00,
     because BEQ works fine here.
FBCF  Save higher page byte in X-register (DD00)
FBD2  Get lower page byte in A-reg and (DC00)
FBD5  store A-reg in higher page byte. (DD00)
FBD8  Get X-register and
FBD9  put it in lower page byte. (DC00)
FBCD  RETURN

FBDD  **************************** ALLOCATE I/O BUFFER ****************************

FBDD  ---
FBDF  get I/O buffer page number
FBE2  can't be below $000?
FBE4  else, error >>FC28
FBE6  can't be above $800?
FBE8  else, error >>FC28
FBED  $4A/$4B -- I/O buffer
FBF1  must be page aligned >>FC28
FBF7  ---
FBF8  check each page of I/O buffer for <FC63>
FBFB  prior allocation in system bit map (BF5B)
FC0B  ---
FC09  if ok, mark each Page as allocated <FC63>
FC0C  in system memory bit map (BF5B)
FC19  assign buffer number (REFNUM*2) in FCB (DB00)
FC21  and save buffer location in buffer list
FC26  exit
FC27  RETURN
FC28  bad I/O buffer error
FC2B  RETURN
**ProDOS MLI -- V1.3 -- 2 DEC 86**  
NEXT OBJECT ADDR: FC2C

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

---

**FC2C**  
********** LOCATE I/O BUFFER ***********************  
---

**FC2C**  
---

**FC2D**  
AREG contains buffer number *2 (BF6E)

**FC30**  
move buffer pointer to NXTBUF variable (FED1)

**FC39**  
exit

---

**FC3A**  
********** FREE I/O BUFFER ***********************  
---

**FC3A**  
is buffer already free? <FC2C>

**FC3F**  
yes, exit >>PC61

**FC43**  
zero its address in system global page (BF6F)

**FC50**  
---

**FC51**  
free each page in buffer <FC63>

**FC54**  
by marking system bit map

**FC61**  
exit

**FC62**  
RETURN

---

**FC63**  
********** LOCATE BIT MAP POSITION ***********************  
(GIVEN PAGE NUMBER)

**FC63**  
XREG contains page number

**FC64**  
compute page number times 8

**FC67**  
use as offset for bitmask (PDF4)

**FC6E**  
page number / 8 = byte offset

**FC6F**  
into bitmap

**FC71**  
exit

---

**FC72**  
********** CHECK BUFFER VALIDITY ***********************  
START > $200  END < $BF00

**FC72**  
get buffer address (MSB)

**FC76**  
must be >$200 else error >>PC28

**FC78**  
get length (FECF)

**FC7E**  
compute last page no. of buffer

**FC81**  
---

**FC8A**  
may not extend into $BF00

**FC8C**  
else, error >>PC28

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

---

**FC8F**  
---

**FC90**  
see if this page is allocated <PC63>

**FC96**  
if so, error >>FC28

**FC98**  
else, check other page also

**FC9C**  
then exit if both have been checked

**FC9D**  
RETURN

---

**FC9E**  
********** MLI GET BUFF CALL **********  
******

**FC9E**  
get next available buffer

**FCB3**  
put its address in caller's parm list

**FCAB**  
and exit

**FCAC**  
RETURN

---

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

**FCAD**  
mark his buffer allocated

**FCB2**  
error? >>FCD4

**FCB4**  
get old buffer address (FED2)

**FCBE**  
free old buffer's pages in map <FC49>

**FC85**  
copy old buffer contents

**FCC7**  
to new buffer

**FCD3**  
then exit

**FCDA**  
RETURN

---

**FCD5**  
********** GO TO QUIT CODE HANDLER ***********************  
---

**FCD5**  
enable 2nd 4K bank of language card (C8H3)

**FCDB**  
(Quit code lives at $D100-$D3FF) (C8H3)

**FCDD**  
get first four bytes of page 8, (0000)

**FCE0**  
save them on the stack

**FCE4**  
Set ($V0) -> $D100

**FCE6**  
Set ($V2) -> $1000

**FCE2**  
Set Y = 0

**FCF3**  
3 pages of code to copy

**FCF5**  
---

**FCF6**  
copy quit code handler to $1000

**FD04**  
pull 4 saved bytes off stack

**FD8D**  
copy HIGH RAM BANK1 (C88B)

**FD1B**  
(MLI) (C88B)

**FD15**  
point RESET vector at $1000 (03F2)

**FD1D**  
set power-up byte properly

**FD22**  
go to quit code handler at $1000 >>1000

---

**FD25**  
********** ACCESS RAM-BASED DEVICE DRIVER ***********************  
This (undocumented?) routine allows a device driver  
to reside in BANK2 of auxiliary high RAM (they normally  
reside in slot ROM). When the device driver is set up,  
the address of this routine, which may be found at $3E4,  
becomes the address of the device driver. Bytes $3E4 and $3E5  
are changed to the address of the real driver in aux high RAM.  
This routine must call the page 3 routine at $3D6  
because the MLI is in main high RAM and will be
<table>
<thead>
<tr>
<th>Addr</th>
<th>Description/Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>FD5E</td>
<td>DEALLOC INTERRUPT</td>
</tr>
<tr>
<td>FD60</td>
<td>UNUSED</td>
</tr>
<tr>
<td>FD61</td>
<td>READ BLOCK</td>
</tr>
<tr>
<td>FD62</td>
<td>WRITE BLOCK</td>
</tr>
<tr>
<td>FD63</td>
<td>GET TIME</td>
</tr>
<tr>
<td>FD64</td>
<td>EXIT</td>
</tr>
<tr>
<td>FD65</td>
<td>CREATE</td>
</tr>
<tr>
<td>FD66</td>
<td>DESTROY</td>
</tr>
<tr>
<td>FD67</td>
<td>RENAME</td>
</tr>
<tr>
<td>FD68</td>
<td>SET FILE INFO</td>
</tr>
<tr>
<td>FD69</td>
<td>GET FILE INFO</td>
</tr>
<tr>
<td>FD6A</td>
<td>ON LINE</td>
</tr>
<tr>
<td>FD6B</td>
<td>SET PREFIX</td>
</tr>
<tr>
<td>FD6C</td>
<td>GET PREFIX</td>
</tr>
<tr>
<td>FD6D</td>
<td>OPEN</td>
</tr>
<tr>
<td>FD6E</td>
<td>NEWLINE</td>
</tr>
<tr>
<td>FD6F</td>
<td>READ</td>
</tr>
<tr>
<td>FD70</td>
<td>WRITE</td>
</tr>
<tr>
<td>FD71</td>
<td>CLOSE</td>
</tr>
<tr>
<td>FD72</td>
<td>FLUSH</td>
</tr>
<tr>
<td>FD73</td>
<td>SET MARK</td>
</tr>
<tr>
<td>FD74</td>
<td>GET MARK</td>
</tr>
<tr>
<td>FD75</td>
<td>UNUSED</td>
</tr>
<tr>
<td>FD76</td>
<td>ROUTE EOP</td>
</tr>
<tr>
<td>FD77</td>
<td>GET EOF</td>
</tr>
<tr>
<td>FD78</td>
<td>SET BUF</td>
</tr>
<tr>
<td>FD79</td>
<td>GET BUF</td>
</tr>
<tr>
<td>FD7A</td>
<td>UNUSED</td>
</tr>
<tr>
<td>FD7B</td>
<td>UNUSED</td>
</tr>
<tr>
<td>FD7C</td>
<td>UNUSED</td>
</tr>
<tr>
<td>FD7D</td>
<td>ALLOC INTERRUPT</td>
</tr>
<tr>
<td>FD7E</td>
<td>DEALLOC INTERRUPT</td>
</tr>
<tr>
<td>FD80</td>
<td>UNUSED</td>
</tr>
<tr>
<td>FD81</td>
<td>READ BLOCK</td>
</tr>
<tr>
<td>FD82</td>
<td>WRITE BLOCK</td>
</tr>
<tr>
<td>FD83</td>
<td>GET TIME</td>
</tr>
<tr>
<td>FD84</td>
<td>EXIT</td>
</tr>
<tr>
<td>FD85</td>
<td>CREATE</td>
</tr>
<tr>
<td>FD86</td>
<td>DESTROY</td>
</tr>
<tr>
<td>FD87</td>
<td>RENAME</td>
</tr>
<tr>
<td>FD88</td>
<td>SET FILE INFO</td>
</tr>
<tr>
<td>FD89</td>
<td>GET FILE INFO</td>
</tr>
<tr>
<td>FD8A</td>
<td>ON LINE</td>
</tr>
<tr>
<td>FD8B</td>
<td>SET PREFIX</td>
</tr>
<tr>
<td>FD8C</td>
<td>GET PREFIX</td>
</tr>
<tr>
<td>FD8D</td>
<td>OPEN</td>
</tr>
<tr>
<td>FD8E</td>
<td>NEWLINE</td>
</tr>
<tr>
<td>FD8F</td>
<td>READ</td>
</tr>
</tbody>
</table>
### ProDOS MLI -- V1.3 -- 2 DEC 86

**ADDR** | **DESCRIPTION/CONTENTS**
--- | ---
FD90 | WRITE
FD91 | CLOSE
FD92 | FLUSH
FD93 | SET MARK
FD94 | GET MARK
FD95 | UNUSED
FD96 | SET EOF
FD97 | GET EOF
FD98 | SET BUF
FD99 | ********** MLI COMMAND ADDRESS TABLE **********

| FD99 | CREATE
| FD9B | DESTROY
| FD9D | RENAME
| FD9F | SET FILE INFO
| FDB1 | GET FILE INFO
| FDB3 | ON LINE
| FDB5 | SET PREFIX
| FDB7 | GET PREFIX
| FDB9 | OPEN
| FDBB | NEWLINE
| FDBD | READ
| FDBF | WRITE
| FDC1 | CLOSE
| FDC3 | FLUSH
| FDC5 | SET MARK
| FDC7 | GET MARK
| FDC9 | SET EOF
| FDD1 | GET EOF
| FDD3 | SET BUF
| FDD5 | GET BUF

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

**ADDR** | **DESCRIPTION/CONTENTS**
--- | ---
FDCC | 0 1 0 0 B
FDCD | 0 0 1 0 C
FDCE | 0 0 1 0 D
FDCF | 0 1 0 0 E
FDDO | 0 1 0 0 F
FD1 | 0 1 0 0 10
FD2 | 0 1 0 0 11
FD3 | 0 1 0 0 12
FD4 | 0 1 0 0 13

### FDD5 ********** CONSTANTS - DATA AREA **********

| FDD5 | Blocks Used
| FDD7 | End of File
| FDDA | Special ID (Must be 5 bits on)
| FDDB | 'HUSTOSI' Author's name
| FDE2 | Previous Block of Vol Dir Key Block

---
THE FOLLOWING IS COPIED TO SUBDIR HDR+$20

| FDE4 | Version of ProDOS
| FDE5 | Minimum Version
| FDE6 | Access Byte (D|Rn|B|O00|W|R)
| FDE7 | Entry Length
| FDE8 | Entries per Block
| FDE9 | File Count
| FDEB | Parent LSB (copied to SUBDIR HDR+$20)

---
| FDEC | File Type (Directory)
| FDED | Block Number
| FDEF | Number of Blocks
| FDF1 | End of File

### FDF4 ********** BITMASK TABLE **********

| FDF4 | 00000000
| FDF5 | 01000000
| FDF6 | 00100000
| FDF7 | 00010000
| FDF8 | 00001000
| FDF9 | 00000100
| FDFA | 00000010
| FDFB | 00000001

### FDFC ********** OFFSETS INTO FILE CONTROL BLOCKS **********

| FDFC | FCB's are at $D800-$D8FF

### FDFC Key Block
Beneath Apple ProDOS Supplement

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

ADDRESS/DESCRIPTION CONTENTS

FDPE # Blocks Used
FES0 End of File

FES3 ********** SET/GET FILE_INFO OFFS
FES3 Access
FES4 File Type
FES5 Aux Type
FES7 Storage Type
FES8 Blocks Used (MSB on means GET)

FES8A Datetime (Last Mod)
FES8E Datetime (Creation)

FE12 ********** FATAL ERROR MESSAGE **

FE12 ' INSERT SYSTEM DISK AND RESTART '

FEX3 ***

FEX3A ********** VARIABLES - DATA AREA

FEX3A Parent Pointer Block
FEX3C Parent Entry Number
FEX3D Parent Entry Length
FEX3E Datetime (Creation)
FEX42 Version
FEX43 Min Version
FEX44 Access Byte
FEX45 Entry Length
FEX46 Entries per Block
FEX47 File Count
FEX49 Bit Map Pointer
FEX4B Total Blocks

THE FOLLOWING 6 BYTES UNIQUELY IDENTIFY A FILE:

FEX4D Device Number
FEX4E Current Directory Block Number
FEX50 Block Number of File Entry in
FEX52 File Entry Number in Directory

FEX53 ********** FILE ENTRY BUFFER ****

FEX53 Type/Length (TTTTLLLL)
FEX54 File Name (Max 15) >0000F
FEX63 File Type
FEX64 Key Pointer
FEX66 Blocks Used

FEX66 Bit Map Offset in
FEX67 Bit Map Page Offset
FEX68 Bit Map Buffer Page (if > 0)
FEX69 Bit Map Flag (if 88, no
FEX9A Bit Map DEVNUM
FEX9B Bit Map Block Number
FEX9D Bit Map Block offset fo

FREE FCB's refnum
(if > 0) or...
with free bit on or...

0 to search

FEX82 Directory Header
FEX8C Entry Header

FEX89 Page

(1)

FEX80 Page

(1)

FEX89 Page

(1)

Multiblock Bitmaps

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

ADDRESS/DESCRIPTION CONTENTS

FE66 End of File
FE6B Datetime (Creation)
FE6F Version
FE70 Min Version
FE71 Access Attribute
FE72 Aux Type (Load Address/FE74 Datetime (Last Mod)

FE77 Header Pointer

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

FE7A 3 Byte Scratch

FE7D ---

FE7E End of File

FE81 Previous Mark

FE84 Compare Vol Name Scratch
FE85 Offset into VCB Table
FE86 Offset into FCB Table
FE87 Free FCB found Flag

FE88 Number of Free Blocks

FE8A Storage Type

FE8B FCB already open flag

FE8C File Count

FE8E Entries/Block Loop Count

Free Entry Found Flac

FE8F bit for free

FE90 # Blocks inBitmap left

FE91 Y Register temp

FE92 Pathname Length

FE93 Devnum for Prefix Direc-

FE94 Block of Prefix Direc-
New Mark to be Positioned to for Set Mark
or New Moving Mark (for READ)

FE9E or New EOF for SET_EOF

FEA1 Request Count (Read/Write etc.)
FEA3 Multi-Block I/O count
FEA4 Newline character
FEA5 Newline mask
FEA6 I/O Transfer occurred flag
FEA7 MLI Command * 2
FEA8 ORed into Access Flags ($28 - Backup)
FEA9 Duplicate Volume Flag (if $FF)
FEAA Duplicate Volume's VCB index
FEAB MLI function code (low 5 bits)
Characters in current Pathname index lvl or
FEAC ONLINE: volname len - loop index
FEAD new pathname: index to last name
old pathname: index to last name or...
FEAE ONLINE: index to data buffer
FEAF Old PFI XPTR value
FEB0 Pathname fully qualified flag (if $FF)
Pathname: temp save area for index or...

FEB1 ONLINE: DEVCNT
FEB2 close-all error code
FEB3 Set EOF: new Key Block pointer
FEB5 New storage type (SET_EOF)
FEB6 Freed Blocks count
FEB8 EOF Block number (MSB then LSB)
FEBA EOF byte offset into Block
FEBC EOF - Master index counter
FEBD Save area for index into table below

FEBE *********** DEVICE TABLE BUILT BY ONLINE ****************
(also used by SET_EOF to keep track of
8 blocks to be freed at a time)

FECE device table part one
FECD device table part two

FEE1 length of path, etc.
FED1 next buffer address
FED3 16 byte stack save area
FEE3 6 byte zero page save area

FEE9 ---
FEE9 Jump Vector, used for indirect jumps
FEEB Destroy flag (1 = destroy operation)
**ProDOS System Global Page -- V1.2**

**ADDR** | **DESCRIPTION/CONTENTS**
---|---
BF50 | **MODULE STARTING ADDRESS**

* * PRODOS GLOBAL PAGE *
* *
* VERSION 1.2 -- 6 SEP 86 *
* VERSION 1.2 -- 2 DEC 86 *
* *

**BF00********** MLI AND IRQ HANDLER EQUATES ****************************

DE00 | Main MLI entry point.
DF0C | Address for no device connected.
DF4E | IRQ handler within MLI.
DFFF | System error handler.
E009 | System death handler.
FF08 | Patch in ProDOS IRQ Handler.

**BF00********** JUMP VECTORS ****************************

BF00 | ENTRY 
BF03 | JMP to MLI. >>BF4B
BF64 | JMP to IRQ handler. >>BF03
BF66 | JMP To Date/Time routine (MTS if no clock).
BF67 | Normal clock code address.
BF69 | JMP to system error handler. >>DF0F
BF6C | JMP to system death handler. >>E009
BF6F | SERR 

**BF10********** DEVICE INFORMATION ****************************

BF10 | DEVADK01 Slot 0 reserved.
BF12 | DEVAK11 Slot 1, drive 1 device driver address.
BF14 | DEVAK21 Slot 2, drive 1 device driver address.
BF16 | DEVAK31 Slot 3, drive 1 device driver address.
BF18 | DEVAK41 Slot 4, drive 1 device driver address.
BF1A | DEVAK51 Slot 5, drive 1 device driver address.
BF1C | DEVAK61 Slot 6, drive 1 device driver address.
BF1E | DEVAK71 Slot 7, drive 1 device driver address.
BF20 | DEVADK02 Slot 0 reserved.
BF22 | DEVAK12 Slot 1, drive 2 device driver address.
BF24 | DEVAK22 Slot 2, drive 2 device driver address.
BF26 | DEVAK32 Slot 3, drive 2 device driver address.
BF28 | DEVAK42 Slot 4, drive 2 device driver address.
BF2A | DEVAK52 Slot 5, drive 2 device driver address.
BF2C | DEVAK62 Slot 6, drive 2 device driver address.
BF2E | DEVAK72 Slot 7, drive 2 device driver address.

**BF20********** PRODOS GLOBAL PAGE -- V1.2**

**ADDR** | **DESCRIPTION/CONTENTS**
---|---
BF20 | DEVAK01 Slot and drive (DBSSS0008) of last device.
BF21 | DEVAK0NT Count (minus 1) of active devices.
BF22 | DEVAKLST List of active devices (slot, drive, and identification=USSSII11).
BF40 | "(C)APPLE'83" Copyright notice.
BF4B | MLIEN1 Push status reg on stack.
BF4C | Disable interrupts.
BF4D | Go to MLIEN1. >>BF27
BF50 | APTIQ Read high RAM, BANK1 (C088) and continue IRQ exit. >>FF08
BF58 | BITMAP Bitmap of low 48K of memory.
BF70 | BUFFER1 Open file 1 buffer address.
BF72 | BUFFER2 Open file 2 buffer address.
BF74 | BUFFER3 Open file 3 buffer address.
BF76 | BUFFER4 Open file 4 buffer address.
BF78 | BUFFER5 Open file 5 buffer address.
BF7A | BUFFER6 Open file 6 buffer address.
BF7C | BUFFER7 Open file 7 buffer address.
BF7E | BUFFER8 Open file 8 buffer address.

**BF80********** INTERRUPT INFORMATION ****************************

BF90 | INTRK1 Interrupt handler address (highest priority).
BF92 | INTRK2 Interrupt handler address.
BF94 | INTRK3 Interrupt handler address.
BF96 | INTRK4 Interrupt handler address (lowest priority).
BF98 | INTREG A-register savearea.
BF99 | INTREG X-register savearea.
BF9A | INTREG Y-register savearea.
BF9B | INTREG S-register savearea.
BF9C | INTREG F-register savearea.
BF9D | INTRKID Bank ID byte (ROM, RAM1, or RAM2).
BF9E | INTADDR Interrupt return address.

**BF90********** GENERAL SYSTEM INFO ****************************

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

00... 0... II
Beneath Apple ProDOS Supplement

ProDOS System Global Page -- V1.2

ADDR DESCRIPTION/CONTENTS

---

01. 0... II+
10. 0... Ile or IIGS
11. 0... Future expansion
00. 1... Future expansion
01. 1... Future expansion
10. 1... Iic
11. 1... Future expansion
00... Unused
01... 48K
10... 64K
11... 128K
X... Reserved
0... No 80-column display
1... 80-column display
0... No compatible clock
1... 1 Compatible clock present

BF99 SLTBYT  Slot ROM map (bit on indicates ROM present)
BF9A PPFIXPTR Prefix flag (0 indicates no active prefix)
BF9B MLIACVT  MLI active flag (1 indicates active)
BF9C CMDADR  Last MLI call return address
BF9E SAVEX Y-register save area for MLI calls
BF9F SAVEX  Y-register save area for MLI calls

BFA0 ********** HANDLE BANK SWITCHING AFTER IRQ ***********

(Enter reading high RAM, BANK1)

BFA0 EXIT $E000 same as save byte? (E0W0)
BFA3 Yes, check for BANK1/BANK2 >>BFAA
BFA5 No, enable ROM (C0B2)
BFA8 and exit now. >>BF95
BFAA EXIT1 Get RAM save byte for $D000. (BF95)
BFAF Is it the same as BANK1? (D000)
BFB0 Yes, exit now. >>BF95
BFB2 No, switch to BANK2. (C0B3)
BFB5 EXIT2 Restore A-Register and return from the interrupt.

BFB7 ******** MLI ENTRKY, CONTINUED *********************

BFB7 MLICONT Carry set will
BFB8 rroll flag bit into MLIACVT. (BF98)
BFB8 By storing $E000 in BNKBYT1 (BF4)
BFC1 and $D000 (D000)
BFC4 in BNKBYT2. (BF5)
BFC7 Then enable high RAM, BANK1 (C0B8)
BFC8 for read and write. (C0B8)
BFCD Jump to actual MLI. >>DE00

BFD0 ********** INTERRUPT ********** ROUTINES **********

BFD0 IRXKIT Data
BFD3 HWKIME state of high memory. (BF8D)
BFD5 High RAM, BANK1 enabled. >>BFE2
BFD6 SYSH RAM, BANK2 enabled. >>BFDF
BFD8 Yes;em have only 48K
BFD9 Sane only ROM in high memory. >>BFE7
BFD2 A whole ROM. (C0B1)
BFD3 IRXKIT1 Swiifs branch. >>BFE7
BFE2 IRXKIT2 Presch to BANK2. (C0B3)
BFE4 Latent BANKID for ROM
BFE7 ROMKIT Reset reset if high RAM interrupt. (BF8D)
BFEA exiore accumulator and (BF88)
BFEB IRQENF Enable
BFE9 FW:ile high RAM, BANK1 (C0B8)
BFF1 Jumpread and write. (C0B8)
BFF4 ********** DATA **********

BFF4 BNKBYT1 Storage
BFF5 BNKBYT2 Storage for byte at $E000.
BFF6 BNKBYT6 Storage for byte at $D000.
$9-$BF95 currently not used.

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

BFFC IBVKVER High
BFFD IVERSION Version number of Kernel needed for this interpreter.
BFFE KBKVER Current number of this in interpreter.
BFFF KVERSION MLIently undefined. Reserved for future use.

=2 Version number:
=3 in Version 1.2
=4 in Version 1.3
### Beneath Apple ProDOS Supplement

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

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

<table>
<thead>
<tr>
<th>1000</th>
<th><strong>QUIT Code</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>STORED in BANK 2 of High RAM</strong></td>
<td></td>
</tr>
<tr>
<td>A move routine in the MLI moves the code to $1000 and JMPs to it when a QUIT command is issued.</td>
<td></td>
</tr>
<tr>
<td><strong>VERSION 1.2 -- 6 SEP 86</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Move routine at $FCA9</strong></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1001</th>
<th><strong>VERSION 1.3 -- 2 DEC 86</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Move routine at $FC05</strong></td>
<td></td>
</tr>
</tbody>
</table>

| **0024** | Cursor Horizontal |
| **0025** | Cursor Vertical |

<table>
<thead>
<tr>
<th><strong>1000</strong></th>
<th><strong>EXTERNAL EQUATES</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>0200</strong></td>
<td>Prefix Buffer</td>
</tr>
<tr>
<td><strong>0200</strong></td>
<td>Buffer</td>
</tr>
<tr>
<td><strong>BF00</strong></td>
<td>MLI Entry</td>
</tr>
<tr>
<td><strong>BF58</strong></td>
<td>Bitmap</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>1000</strong></th>
<th><strong>SOFTWARE SWITCHES</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>C000</strong></td>
<td>Keyboard</td>
</tr>
<tr>
<td><strong>C00C</strong></td>
<td>Disable 80 column card</td>
</tr>
<tr>
<td><strong>C00E</strong></td>
<td>Select standard character set</td>
</tr>
<tr>
<td><strong>C082</strong></td>
<td>ROM select</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>1000</strong></th>
<th><strong>MONITOR EQUATES</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>FE84</strong></td>
<td>Initialize 40-column display</td>
</tr>
<tr>
<td><strong>FC58</strong></td>
<td>Home</td>
</tr>
<tr>
<td><strong>FC9C</strong></td>
<td>Clear to end of line</td>
</tr>
<tr>
<td><strong>FD0C</strong></td>
<td>Read a key</td>
</tr>
<tr>
<td><strong>FD8E</strong></td>
<td>Output a Carriage Return</td>
</tr>
<tr>
<td><strong>FEDD</strong></td>
<td>Output a Character</td>
</tr>
<tr>
<td><strong>FEB4</strong></td>
<td>Set Normal display</td>
</tr>
<tr>
<td><strong>FE93</strong></td>
<td>Set Video</td>
</tr>
</tbody>
</table>

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

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

<table>
<thead>
<tr>
<th><strong>1000</strong></th>
<th><strong>INITIALIZATION</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1000</strong></td>
<td>Select ROM (C082)</td>
</tr>
<tr>
<td><strong>1001</strong></td>
<td>Disable 80 column card (CWAC)</td>
</tr>
<tr>
<td><strong>1006</strong></td>
<td>Select standard character set (C00E)</td>
</tr>
<tr>
<td><strong>1009</strong></td>
<td>Clear 80-column store (C000)</td>
</tr>
<tr>
<td><strong>108C</strong></td>
<td>Set Normal display (white on black) (FE84)</td>
</tr>
<tr>
<td><strong>108F</strong></td>
<td>Initialize 40-column display (FE84)</td>
</tr>
<tr>
<td><strong>1082</strong></td>
<td>Set Video (FE93)</td>
</tr>
<tr>
<td><strong>1085</strong></td>
<td>Set Keyboard (FE89)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>1018</strong></th>
<th><strong>INITIALIZE MEMORY BITMAP</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1018</strong></td>
<td>Mark pages $00, $1, $4 through $7</td>
</tr>
<tr>
<td><strong>101A</strong></td>
<td>and $BF as in use</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>102D</strong></th>
<th><strong>DISPLAY CURRENT PREFIX</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>102D</strong></td>
<td>Clear Screen and Home cursor (FC58)</td>
</tr>
<tr>
<td><strong>1030</strong></td>
<td>Go down 1 line (FD08)</td>
</tr>
<tr>
<td><strong>1033</strong></td>
<td>Point to prompt number 1</td>
</tr>
<tr>
<td><strong>1035</strong></td>
<td>and print it out (FD06)</td>
</tr>
<tr>
<td><strong>1038</strong></td>
<td>Position to line 3</td>
</tr>
<tr>
<td><strong>103F</strong></td>
<td>Call MLI (GET PREFIX) (BF00)</td>
</tr>
<tr>
<td><strong>1042</strong></td>
<td>Data: GET PREFIX command number</td>
</tr>
<tr>
<td><strong>1043</strong></td>
<td>Data: Pointer to Parameter list</td>
</tr>
<tr>
<td><strong>1045</strong></td>
<td>Get length of Prefix (BF00)</td>
</tr>
<tr>
<td><strong>104A</strong></td>
<td>Put a 0</td>
</tr>
<tr>
<td><strong>104A</strong></td>
<td>at the end of the Prefix (BF01)</td>
</tr>
<tr>
<td><strong>104D</strong></td>
<td>Check prefix length... (BF00)</td>
</tr>
<tr>
<td><strong>1050</strong></td>
<td>If length=0, there is no current Prefix &gt;&gt;105D</td>
</tr>
<tr>
<td><strong>1052</strong></td>
<td>If non-zero, display the current Prefix (BF00)</td>
</tr>
<tr>
<td><strong>1057</strong></td>
<td>on the video screen (BF5F)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>105D</strong></th>
<th><strong>GET PREFIX NAME</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>105D</strong></td>
<td>Initialize counter</td>
</tr>
<tr>
<td><strong>1064</strong></td>
<td>Read a key (FD0C)</td>
</tr>
<tr>
<td><strong>1067</strong></td>
<td>Is it CARRIAGE RETURN?</td>
</tr>
<tr>
<td><strong>1069</strong></td>
<td>Yes, then accept Prefix &gt;&gt;10BD</td>
</tr>
<tr>
<td><strong>106A</strong></td>
<td>No, then save character</td>
</tr>
<tr>
<td><strong>106C</strong></td>
<td>Clear to end of line (FC9C)</td>
</tr>
<tr>
<td><strong>106F</strong></td>
<td>Retrieve character</td>
</tr>
<tr>
<td><strong>1070</strong></td>
<td>Is it ESCAPE?</td>
</tr>
<tr>
<td><strong>1072</strong></td>
<td>Yes, start all over again &gt;&gt;102D</td>
</tr>
<tr>
<td><strong>1074</strong></td>
<td>Is it CANCEL?</td>
</tr>
<tr>
<td><strong>1076</strong></td>
<td>Yes, start all over again &gt;&gt;102D</td>
</tr>
<tr>
<td><strong>1078</strong></td>
<td>Is it TAB?</td>
</tr>
<tr>
<td><strong>107A</strong></td>
<td>Yes, sound Bell, get another character &gt;&gt;1093</td>
</tr>
</tbody>
</table>
**DESCRIPTION/CONTENTS**

1025 Is it DELETE?
1026 Yes, >>1084
1027 No, keep checking >>1091
1028 Yes, is there room to move back?
1029 No, don't try >>1088
1030 Decrement cursor horizontal position
1031 Decrement counter
1032 Clear to end of line <PC9C>
1033 Try again >>1064
1034 Continue if greater or equal to BACKSPACE >>1099
1035 Else, sound Bell <PF3A>
1036 Try again >>1064
1037 Is it less than or equal to "z"?
1038 Yes, keep checking >>109F
1039 Turn off lowercase
1040 Is it less than ","?
1041 No, Invalid - try again >>1093
1042 Is it greater than "Z"?
1043 Yes, Invalid - try again >>1093
1044 Is it less than or equal to "g"?
1045 Yes, keep checking >>10AF
1046 Is it less than "a"?
1047 Yes, Invalid - try again >>1093
1048 Else, valid character - increment counter
1049 Found 39 characters?
1050 Yes, then start all over >>1876
1051 Put valid character in buffer (0280)
1052 and Print it <FD0E>
1053 Go back for more >>1064
1054 Check counter
1055 If 0 then go on >>10D3
1056 Else, save length (0280)
1057 Call MLI (GET_PREFIX) <<BF00>
1058 Data: SET_PREFIX command number
1059 Data: Pointer to Parameter list
1060 Carry on if no error >>10D3
1061 Sound Bell <PF3A>
1062 Force branch to
1063 always be taken >>1076

********** GET APPLICATION NAME **********

1064 Clear Screen and Home cursor <PC5A>
1065 Go down 1 line <FD0E>
1066 Point to prompt number 2
1067 and print it out <1100>
1068 Position to line 3

**ADATA**

**ProDOS**

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

**NEXT OBJECT ADDR: 107C**

**ADDRESS**

**DESCRIPTION**

1025 Initialize counter
1027 Wait for keypress <FD0C>
1029 Is it ESCAPE?
102E No, keep checking >>10F4
102B Yes, get Cursor horizontal position
10F0 If no, try again >>10D3
10F2 If 0, start all over again >>10D1
10F4 Is it CANCEL?
10F6 Yes, try again >>10D3
10FB Is it -A?
10FA Yes, sound Bell - try again >>1109
10FC Is it DELETE?
10FE Yes >>104
1100 Is it BACKSPACE?
1102 No, keep checking >>11D7
1104 Yes, then handle it >>11C0
1107 Continue if greater than BACKSPACE >>110F
1109 Sound Bell <PF3A>
110C Go back and try again >>1087
1110 Is it CARRIAGE RETURN?
1111 Yes, then go load Application >>113C
1113 Is it less than or equal to "Z"?
1115 Yes, keep checking >>1119
1117 Turn off lowercase
1119 Is it less than ","?
111B Yes, Invalid - try again >>1109
111D Is it greater than "Z"?
111F Yes, Invalid - try again >>1109
1121 Is it less than or equal to "g"?
1123 Yes, keep checking >>1129
1125 Is it less than "a"?
1127 Yes, Invalid - try again >>1109
1129 Else, valid character - save it
112A Clear to end of line <PC9C>
112D Retrieve character
112E and Print it <FD0E>
1131 Increment counter
1132 Found 39 characters?
1134 Yes, start again >>10F6
1136 No, get character in buffer (0280)
1139 and get another >>1087

113C LOAD AND EXECUTE APPLICATION

113C Output a blank
1141 Store length of Application name (0280)
1144 Call MLI (GET_FILE_INFO) <<BF00>
1147 Data: GET_FILE_INFO command number
1148 Data: Pointer to Parameter list

******* GET APPLICATION NAME **********
Beneath Apple ProDOS Supplement

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

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>114A</td>
<td>Continue if no error &gt;&gt;114F</td>
</tr>
<tr>
<td>114C</td>
<td>Else, go to Error Handler &gt;&gt;11E2</td>
</tr>
<tr>
<td>114F</td>
<td>Get File Type (12A5)</td>
</tr>
<tr>
<td>1152</td>
<td>Is it ProDOS System file?</td>
</tr>
<tr>
<td>1154</td>
<td>Yes, continue &gt;&gt;115B</td>
</tr>
<tr>
<td>1156</td>
<td>No, indicate Error $01</td>
</tr>
<tr>
<td>1158</td>
<td>Go to Error Handler &gt;&gt;11E2</td>
</tr>
<tr>
<td>115B</td>
<td>Set Reference number to 0</td>
</tr>
<tr>
<td>1160</td>
<td>Call MLI (CLOSE) &lt;BF00&gt;</td>
</tr>
<tr>
<td>1163</td>
<td>Data: CLOSE command number</td>
</tr>
<tr>
<td>1164</td>
<td>Data: Pointer to Parameter list</td>
</tr>
<tr>
<td>1166</td>
<td>Continue if no error &gt;&gt;1168</td>
</tr>
<tr>
<td>1168</td>
<td>Else, go to Error Handler &gt;&gt;11E2</td>
</tr>
<tr>
<td>116B</td>
<td>Get Access Byte (12A9)</td>
</tr>
<tr>
<td>1170</td>
<td>Yes, &gt;&gt;1177</td>
</tr>
<tr>
<td>1172</td>
<td>No, Indicate Error $27</td>
</tr>
<tr>
<td>1174</td>
<td>Go to Error Handler &gt;&gt;11E2</td>
</tr>
<tr>
<td>1177</td>
<td>Call MLI (OPEN) &lt;BF00&gt;</td>
</tr>
<tr>
<td>117A</td>
<td>Data: OPEN command number</td>
</tr>
<tr>
<td>117B</td>
<td>Data: Pointer to Parameter list</td>
</tr>
<tr>
<td>117D</td>
<td>Continue if no error &gt;&gt;1182</td>
</tr>
<tr>
<td>117F</td>
<td>Else, go to Error Handler &gt;&gt;11E2</td>
</tr>
<tr>
<td>1182</td>
<td>Get Reference Number (12B8)</td>
</tr>
<tr>
<td>1185</td>
<td>and update READ and (12BC)</td>
</tr>
<tr>
<td>118B</td>
<td>GET EOF parameter lists (12C4)</td>
</tr>
<tr>
<td>1189</td>
<td>Call MLI (GET EOF) &lt;BF00&gt;</td>
</tr>
<tr>
<td>118E</td>
<td>Data: GET_EOF command number</td>
</tr>
<tr>
<td>118F</td>
<td>Data: Pointer to Parameter list</td>
</tr>
<tr>
<td>1191</td>
<td>If error, handle it &gt;&gt;11E2</td>
</tr>
<tr>
<td>1193</td>
<td>Is EOF mark less than $10000 (12C7)</td>
</tr>
<tr>
<td>1196</td>
<td>Yes, continue on &gt;&gt;119C</td>
</tr>
<tr>
<td>1198</td>
<td>No, Indicate Error $27</td>
</tr>
<tr>
<td>119A</td>
<td>Go to Error Handler &gt;&gt;11E2</td>
</tr>
<tr>
<td>119C</td>
<td>Transfer EOF to Request count (12C5)</td>
</tr>
<tr>
<td>119F</td>
<td>in READ parameter list (12BF)</td>
</tr>
<tr>
<td>11A0</td>
<td>Call MLI (READ) &lt;BF00&gt;</td>
</tr>
<tr>
<td>11A8</td>
<td>Data: READ command number</td>
</tr>
<tr>
<td>11AC</td>
<td>Data: Pointer to Parameter list</td>
</tr>
<tr>
<td>11AE</td>
<td>Save status of READ</td>
</tr>
<tr>
<td>11AF</td>
<td>Call MLI (CLOSE) &lt;BF00&gt;</td>
</tr>
<tr>
<td>11B2</td>
<td>Data: Get Prefix command number</td>
</tr>
<tr>
<td>11B3</td>
<td>Data: Pointer to Parameter list</td>
</tr>
<tr>
<td>11B5</td>
<td>Continue if no error &gt;&gt;11BB</td>
</tr>
<tr>
<td>11B7</td>
<td>Else, retrieve status</td>
</tr>
<tr>
<td>11BB</td>
<td>and go to Error Handler &gt;&gt;11E2</td>
</tr>
</tbody>
</table>

This area of the code was modified for Version 1.2, and a bug was fixed for the P-register is non-zero; may not force the required label there that was READ good? |

11C9 Get cursor position horizontal Get EOF parameter lists (12C4) |
11C9 Get cursor position vertical |
11C9 Get cursor position vertical Get cursor back 2 spaces |
11C9 Get cursor position vertical Output a space |
11C9 Get cursor position vertical Move cursor back 1 space |
11C9 Get cursor position vertical Return to get another character >>10E7 |
11C9 Get cursor position horizontal Move cursor back 1 space |
11C9 Get cursor position horizontal Move cursor back 1 space |
11C9 Get cursor position horizontal Return to caller |

11D2 SAVE ERROR MESSAGE

11D2 Save Accumulator (Error number) |
11E4 Position to line 12 "ber" |
11EB Get Error number |
11ED Is it 1? |
11EF No, keep checking >>11PS |
11F1 Yes, point to error message |
11F3 and go print it >>120B |
11F5 Is it $40? |
11F7 Yes, print error message 3 |
11F9 Is it $444? |
11FB Yes, print error message 3 |
11FD Is it $45? |
11FF Yes, print error message 3 |
1201 Is it $46? |
1203 Yes, print error message 3 |
1205 Point to error message 2 |
1207 and go print it >>120B |
11D2 SAVE ERROR MESSAGE
ProDOS QUIT Code -- V1.2 -- 6 SEP 86

--- ADDRESS --- DESCRIPTION/CONTENTS --- NEXT OBJECT ADDR: 12BA

1209 Point to
120B Print...
120E Get app...

--- DESCRIP... --- NEXT OBJECT ADDR: 1209

1211 ************

Prompt error message 3
Prompt for message <11D6>
Prompt for location name again >10DE

1239 'ENTER ASCII TEXT **********************

125C Ring Bell
125D 'NOT A PREFIX (PRESS "RETURN" TO ACCEPT)'

1273 Ring Bell
1274 'I/O ERR:

128A Ring Bell
128B 'FILE/FA:

12A1 ************

GET_P...

12A2 Parmcount...
12A4 Access...
12A5 File Typ...
12A6 Aux_Typ...
12A8 Storage...
12A9 Blocks...
12AB Date...
12AF Datetime...

--- OPEN PE ---

12B3 Parmcount...
12B4 Pathname...
12B6 1/O Suffix...
12B8 Reference...

--- CLOSE Parmlist ---

12B9 Parmcount...
12BA Reference...

--- READ Parmlist ---

12BB Parmcount...
12BC Reference Number...
12BD Data Buffer...
12BF Request Count...
12C1 Transfer Count...

--- GET_EOF Parmlist ---

12C3 Parmcount...
12C4 Reference Number...
12C5 EOF Mark...

--- GET/SET_PREFI Parmlist ---

12CB Parmcount...
12C9 Pathname...

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

12CB These unused bytes are $D3CB-$D3FF in high RAM
12FF and $58CB-$59FF when loaded as part of "PRODOS" file.
**Disk II Device Supplement**

**ADDR** | **DESCRIPTION** | **NEXT OBJECT ADDR**
--- | --- | ---
D000 | MODULE | D000

**VER.** | **1.2 -- 6 SEP 86**
**5.2** |
**RESIDES AT $D000 - $D0FF**

**FUNCTIONS:**

- Set Block Number to a Track and Sector
- Block Number to a Track and Sector
- Disk Device Driver
- Execute
- Restore
- Command
- Increment Sector Number
- Get error
- Indicate
- Return

**EQUATIONS:**

- Internal Equations
- Lock Buffer (1st half)
- Lock Buffer (2nd half)
- PHASE
- MOTOR
- Drive
- Set Recorder
- Read Data Register
- Write Data Register

**Comments:**

- Check code in case IWM device in this slot <D6BE>
- 512.5" Disk Driver Entry
- fit up
- Check LOCAL mode
- If not, code below will
- against Table at $D196
- Valid exit with error

**Execution:**

- Execute
- Restore
- Command
- Increment Sector Number
- Get error
- Indicate
- Return

- No error
- Restore Sector Number
- Exit >> D030
- Execute Buffer Pointer
- Decrement Sector Number by 2 for rest of block
- Get error Command <D38>
- Increment Sector Number - Was prior action ok?
- Increment, Exit >> D030
- Execute Buffer Pointer
- Decrement Sector Number by 2 for rest of block
- Get error Command <D38>
- Return no error (start of block) or number (if any - 0 indicates no error) (<D35>)

**Indicate I/O Error Routine**

- Set Car
- Return "I/O Error" by flag

**Set rec VAIN CODE**

- Preserv
- Get "Uncertification count to 1
- Strip of sector number (<D37>)
- Preservitnum
- Check Unlock Drive <D3000>
- See if slot number
- Save test slot change, turn off motor if so <D6B>
- Initial motor and <D4A>
- See if results
- Update slot counter for delay routine (<D37>)
- Save slot or drive has changed (<D35>)
- Put drive "Current" unit number (<D35>)
- Turn motor results
- Select ve number in Carry flag
- Check for on (<D89>)
- Yes, appropriate drive (<C8A)
- Wait for results - Same slot/drive?
- To commit skip delay >> D072
- Is current new Drive
- Yes, in to speed (<D35>)
- Get transid a status request?
- And go (do not move disk arm) >> D07C
- Check ask number for current request (<D35>)
- Yes, there << D10C
- Got results - Was motor on?
- Yes, skip delay >> D08E
Disk II Device Driver

ADDRESS DESCRIPTION

D07F Wait for D
D081 come up to drive
D089 Is motor 0 speed <D085>
D08C No, then Pn yet? <D4A4>
D08E Is command exit with error -->D0BE
D090 Yes, then a "status" request?
D092 Is command determine status -->D0FD
D093 Yes, then a "read" request?
D095 Prepare to continue -->D09B
D099 back for write <preliminate> <D5F0>
D099 Initializie
D09D -- "retry" count at 64 (D369)
D09F Read an ng:
D0A2 Yes, then dress field - Good read? <D39B>
D0A4 Decrement continue on -->D0BE
D0A7 Yes, then "retry" count - More to try? (D369)
D0A9 No, just try again -->D09D
D0AB Decrement a case "I/O Error"
D0AE No, then a "recalibration" count - More to try? (D36A)
D0B0 Get "current" with error -->D0BE
D0B1 Preserve INT track (D35A)
D0B4 Double it to
D0B5 add 16 to
D0B7 Initializie for recalibration
D0BC Branch alway ON
D0C2 Was the Pnys taken -->D0CC
D0C4 Yes, then ght track found? (D35A)
D0C6 Get "current" continue on -->D0D5
D0C8 Preservr INT track (D35A)
D0CA Get track
D0CB Double it if found
D0CC Put new va
D0CF Get track gue in Device Track Table <D4D3>
D0DF And go where the
D0D3 Branch again <D10C>
D0D9 Was the Pnys taken -->D09D
D0DB No, then light sector found? (D357)
D0DF Is commandry again -->D0B4
D0E0 Yes, then a "write" request?
D0E2 Read the to do it -->D0F4
D0E5 No, then data - Good read? <D3FD>
D0EE Indicate 0r error on -->D04A
D0E9 BNE instruction errors
D0EA Indicate active, never taken
D0EB Preserve Error
D0EF Get Slot Error number (D35B)
D0F0 Turn motor
D0F3 Return to off (CU8A)
caller

DISK II Device Driver

ADDRESS DESCRIPTION

D0F4 HANDLE "protect error"
D0F5 Write data
D0F7 Yes, then EXIT-TUS
D0F9 Indicate "write"
D0FB Branch always

D0FD GET SPARE flag

D0FD Get "protect" status (D0BE)
D0F7 Check "write"-
D0F9 Put result in COSEIVED TRARK
D0F1 Select read mode
D0F9 Exit with approx number for proper phase

D10C Device Track Table <D4F1>
D10D Preserve INT track (D35A)
D110 Turn all phase track (D36F)
D113 Get offset of track Table (D359)
D116 Get track (D359)red track <D133>
D119 Update current number, starting with 3
D11C Get destination
D11F Update Device <D18A>
D122 Move arm to dest number - More to do?
D124 Initialize phase until all phases done -->D127
D127 --- "number by 2 (D35A)
D128 Clear a phase
D129 Decrement phase
D12C Yes, then call ROUTINE
D12E Divide track number
D132 Return to call to find (D372)

D133 ADD appropriate phase and exit
D135 Compute delta-tracks (D371)
D136 Are we already to find to compute delta-tracks
D139 Yes, then set where? (D372)
D143 Initialize phase prior phase and exit
D144 Preserve "current" tracks - go move arm out
D146 Subtract track tracks - Get absolute value-delta
D147 Are we already at phase to move in (D35A)
D14A Yes, then clear taken -->D15A
D14C Positive delta-value track less 1
D14E Negative delta at phase to move out (D35A)
D150 Increment current
D153 Branch always
D155 Compute absolute
D157 Decrement current

T ADDR: D0F3
<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>D15A</td>
<td>Compare delta-tracks with phases moved (D36B)</td>
</tr>
<tr>
<td>D15D</td>
<td>Use smaller value for offset to delay tables &gt;&gt;D162</td>
</tr>
<tr>
<td>D162</td>
<td>Are we pointing at last table value yet?</td>
</tr>
<tr>
<td>D164</td>
<td>Yes, then continue to use current offset &gt;&gt;D168</td>
</tr>
<tr>
<td>D166</td>
<td>Else, use new offset</td>
</tr>
<tr>
<td>D167</td>
<td>Set Carry flag for set phase operation</td>
</tr>
<tr>
<td>D168</td>
<td>Set a phase &lt;D187&gt;</td>
</tr>
<tr>
<td>D16B</td>
<td>Get delay value from table (D373)</td>
</tr>
<tr>
<td>D16E</td>
<td>Delay &lt;D385&gt;</td>
</tr>
<tr>
<td>D171</td>
<td>Get prior phase number (D371)</td>
</tr>
<tr>
<td>D174</td>
<td>Clear Carry flag for clear phase operation</td>
</tr>
<tr>
<td>D175</td>
<td>Clear a phase &lt;D18A&gt;</td>
</tr>
<tr>
<td>D178</td>
<td>Get delay value from table (D37C)</td>
</tr>
<tr>
<td>D17B</td>
<td>Delay &lt;D385&gt;</td>
</tr>
<tr>
<td>D17E</td>
<td>Increment phases moved (D36B)</td>
</tr>
<tr>
<td>D183</td>
<td>Delay &lt;D385&gt;</td>
</tr>
<tr>
<td>D187</td>
<td>Get &quot;current&quot; phase number (D35A)</td>
</tr>
<tr>
<td>D18A</td>
<td>Use low two bits only, zero to three - 000000PP</td>
</tr>
<tr>
<td>D18C</td>
<td>Multiply by two and bring in Carry - 000000FF</td>
</tr>
<tr>
<td>D19D</td>
<td>Merge in slot number - 00000000FF</td>
</tr>
<tr>
<td>D1AF</td>
<td>Put in X-reg for following operation</td>
</tr>
<tr>
<td>D190</td>
<td>Toggle appropriate phase (C000)</td>
</tr>
<tr>
<td>D193</td>
<td>Restore slot number to X-reg</td>
</tr>
<tr>
<td>D195</td>
<td>Return to caller</td>
</tr>
</tbody>
</table>

**TABLE 1**

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>D196</td>
<td>Read Translate Table with Preinitialize Bit mask Tables and Epilog Table in unused areas</td>
</tr>
<tr>
<td>D1A0</td>
<td>00000000</td>
</tr>
<tr>
<td>D1A1</td>
<td>00000000</td>
</tr>
<tr>
<td>D1A2</td>
<td>00000000</td>
</tr>
<tr>
<td>D1A3</td>
<td>00000000</td>
</tr>
<tr>
<td>D1A4</td>
<td>Read Translate Bit Mask 2</td>
</tr>
<tr>
<td>D1C0</td>
<td>00000000</td>
</tr>
<tr>
<td>D1C1</td>
<td>00010000</td>
</tr>
<tr>
<td>D1C2</td>
<td>00000000</td>
</tr>
<tr>
<td>D1C3</td>
<td>00110000</td>
</tr>
<tr>
<td>D1C4</td>
<td>Epilog Table ($DE,$AA,$EB)</td>
</tr>
</tbody>
</table>

**TABLE 2**

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1C7</td>
<td>Read Translate Table</td>
</tr>
<tr>
<td>D1E9</td>
<td>Bit Mask 3</td>
</tr>
<tr>
<td>D1E1</td>
<td>00000000</td>
</tr>
<tr>
<td>D1E2</td>
<td>00001000</td>
</tr>
<tr>
<td>D1E3</td>
<td>00000100</td>
</tr>
<tr>
<td>D1E4</td>
<td>Read Translate Table</td>
</tr>
<tr>
<td>D200</td>
<td>Write Translate Table</td>
</tr>
<tr>
<td>D303</td>
<td>Every 4th byte starting at $D203</td>
</tr>
<tr>
<td>D200</td>
<td>Postnibblize Bit mask Tables</td>
</tr>
<tr>
<td>D201</td>
<td>Bit mask 1 (Every 4th byte starting at $D200)</td>
</tr>
<tr>
<td>D202</td>
<td>Bit mask 2 (Every 4th byte starting at $D201)</td>
</tr>
<tr>
<td>D203</td>
<td>Bit mask 3 (Every 4th byte starting at $D202)</td>
</tr>
<tr>
<td>D356</td>
<td>Auxiliary Buffer ($56 bytes) &gt;&gt;0056</td>
</tr>
<tr>
<td>D356</td>
<td>VARIABLE AREA</td>
</tr>
<tr>
<td>D356</td>
<td>Track number</td>
</tr>
<tr>
<td>D357</td>
<td>Sector number</td>
</tr>
<tr>
<td>D358</td>
<td>Error number</td>
</tr>
</tbody>
</table>

**Table Entry**

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>D359</td>
<td>Current Unit</td>
</tr>
<tr>
<td>D35A</td>
<td>Current Track</td>
</tr>
<tr>
<td>D35B</td>
<td>Slot 1, Devices 1 &amp; 2</td>
</tr>
<tr>
<td>D35D</td>
<td>Slot 2, Devices 1 &amp; 2</td>
</tr>
<tr>
<td>D35F</td>
<td>Slot 3, Devices 1 &amp; 2</td>
</tr>
<tr>
<td>D361</td>
<td>Slot 4, Devices 1 &amp; 2</td>
</tr>
<tr>
<td>D363</td>
<td>Slot 5, Devices 1 &amp; 2</td>
</tr>
<tr>
<td>D365</td>
<td>Slot 6, Devices 1 &amp; 2</td>
</tr>
<tr>
<td>D367</td>
<td>Slot 7, Devices 1 &amp; 2</td>
</tr>
</tbody>
</table>
D369  Retry count (initially 64)
D36A  Recalibration count (initially 1)
D36B  Counter for Read Address routine
D36B  Temporary storage for Read Address routine
D36B  Track counter for Arm Move routine
D36C  Checksum computation
D36D  Volume found
D36E  Sector found
D36F  Delay counter (low byte)
D370  Track found
D370  Checksum found
D371  Delay counter (high byte)
D372  Prior Track
D372  Track number for Arm Move routine

D373  PHASEON/PHASEOFF Tables

D373  Phase on table (delays for disk head acceleration)
D37C  Phase off table (delays for disk head deceleration)

D385  WAIT ROUTINE

D385  Wait about 100 times A-register (microseconds)
D387  ---
D392  ---
D397  Return to caller

D398  READ ADDRESS FIELD

D398  Initialize "must find" count at $FCFC
D39D  Increment count (low order byte) - Zero yet?
D39E  No, skip ahead $D45A
D39E  Increment count (high order byte) - Zero yet? (D36B)
D3A3  Yes, exit and indicate Read Error $D3FB
D3A5  Read data register ($C8BC)
D3A5  Loop until data valid $D3A5
D3AA  Is it first address mark ($DA)?
D3AC  No, then increment "must find" count $D39D
D3AB  Delay for data latch to clear
D3A5  Read data register ($C88C)
D3A5  Loop until data valid $D3A5
D3AB  Loop until data valid $D3A5
D3B3  Initialize count for four byte read
D3B5  Read data register ($C88C)
D3B5  Loop until data valid $D3B5
D3B7  Is it third address mark ($96)?
D3C1  No, then see if it's first address mark $D3AA
D3C3  Set Interrupt flag

D3C4  Initialize checksum
D3C9  Read "odd" encoded byte $D3C9
D3CE  Align "odd" bits $D3C9
D3C9  Save for later (D36B)
D3D0  Read "even" encoded byte $D3D0
D3D0  Combine bytes $D3D0
D3D0  Checksum computation (Volume,Track,Sector,Checksum) (D36D)
D3D0  Do checksum computation (D36C)
D3D0  Decrement counter - Finished field yet?
D3E1  No, do some more $D3C6
D3E3  Is checksum computation zero?
D3E4  No, then exit with carry set $D3FB
D3E6  Read data register ($C8BC)
D3E8  Loop until data valid $D3E8
D3E8  Is it first trailing byte ($DA)?
D3ED  No, then exit with carry set $D3FB
D3E8  Loop for data latch to clear
D3F0  Read data register ($C88C)
D3F1  Loop until data valid $D3F1
D3F5  Is it second trailing byte ($DA)?
D3F7  No, then exit with carry set $D3FB
D3F9  Clear the Carry flag (no error)
D3FA  Return to caller
D3FB  Set the Carry flag (error occurred)
D3FC  Return to caller

D3FD  READ DATA (ON THE FLY) ROUTINE

D3FD  Convert slot number to an
D3FE  absolute reference (i.e. $60 -> $2C)
D403  Modify code for current slot number (D45A)
D403  (i.e. $C8BC,X -> $C8EC) (D473)
D40F  Get data buffer pointers
D413  Modify code for current Buffer address (D4AF)
D416  Provides access to top 3rd of Buffer (D4B0)
D41A  Subtract $54 from current address
D41F  Modify code for current address - $54 (D479)
D422  Provides access to middle 3rd of Buffer (D498)
D426  Subtract $57 from current address
D42B  Modify code for current address - $AB (D470)
D42E  Provides access to bottom 3rd of Buffer (D471)
D431  Initialize must find count at $2D
D433  Decrement count - More to do?
D434  No, then exit $D46D
D436  Read data register ($C88C)
D439  Loop until data valid $D439
D43B  Is it last header mark ($D5)?
D43D  No, then try again $D433
D43F  Loop for data latch to clear
D44B  Read data register ($C88C)
### Disk II Device Driver -- V1.2 -- 6 SEP 86

**NEXT OBJECT ADDR: D443**

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>D443</td>
<td>Loop until data valid &gt;&gt;D440</td>
</tr>
<tr>
<td>D445</td>
<td>Is it 2nd header mark ($AA)?</td>
</tr>
<tr>
<td>D447</td>
<td>No, then see if it is 1st header mark &gt;&gt;D43B</td>
</tr>
<tr>
<td>D449</td>
<td>Delay for register to clear</td>
</tr>
<tr>
<td>D44A</td>
<td>Read data register (C08C)</td>
</tr>
<tr>
<td>D44D</td>
<td>Loop until data valid &gt;&gt;D44A</td>
</tr>
<tr>
<td>D44F</td>
<td>Is it 3rd header mark ($AD)?</td>
</tr>
<tr>
<td>D451</td>
<td>No, then see if it is 1st header mark &gt;&gt;D43B</td>
</tr>
<tr>
<td>D453</td>
<td>Initialize offset into data buffer</td>
</tr>
<tr>
<td>D457</td>
<td>Initialize checksum</td>
</tr>
<tr>
<td>D459</td>
<td>Read a data byte (C08C)</td>
</tr>
<tr>
<td>D45E</td>
<td>Translate it (D100)</td>
</tr>
<tr>
<td>D461</td>
<td>Store it in Auxiliary buffer (D256)</td>
</tr>
<tr>
<td>D464</td>
<td>Compute running checksum</td>
</tr>
<tr>
<td>D466</td>
<td>Increment offset - More to do?</td>
</tr>
<tr>
<td>D467</td>
<td>Yes, then continue &gt;&gt;D457</td>
</tr>
<tr>
<td>D469</td>
<td>Reinitialize offset into data buffer</td>
</tr>
<tr>
<td>D46B</td>
<td>Branch always taken &gt;&gt;D472</td>
</tr>
<tr>
<td>D46D</td>
<td>Get carry flag indicating error</td>
</tr>
<tr>
<td>D46E</td>
<td>Return to caller</td>
</tr>
<tr>
<td>D46F</td>
<td>Store byte in Primary buffer (bottom third) (1000)</td>
</tr>
<tr>
<td>D472</td>
<td>Read a data byte (C08C)</td>
</tr>
<tr>
<td>D477</td>
<td>Translate it and merge in (D100)</td>
</tr>
<tr>
<td>D47A</td>
<td>bits from Auxiliary buffer (D256)</td>
</tr>
<tr>
<td>D480</td>
<td>Increment offset - done yet?</td>
</tr>
<tr>
<td>D481</td>
<td>No, then do another &gt;&gt;D46F</td>
</tr>
<tr>
<td>D483</td>
<td>Save last byte for later, no time now</td>
</tr>
<tr>
<td>D484</td>
<td>Strip off last two bits XXXXXXX0</td>
</tr>
<tr>
<td>D486</td>
<td>Reinitialize offset</td>
</tr>
<tr>
<td>D488</td>
<td>Read a byte (C08C)</td>
</tr>
<tr>
<td>D490</td>
<td>Translate it and merge in (D100)</td>
</tr>
<tr>
<td>D492</td>
<td>bits from Auxiliary buffer (D256)</td>
</tr>
<tr>
<td>D496</td>
<td>Store byte in Primary buffer (middle third) (1000)</td>
</tr>
<tr>
<td>D499</td>
<td>Increment offset - done yet?</td>
</tr>
<tr>
<td>D49A</td>
<td>No, then do another &gt;&gt;D488</td>
</tr>
<tr>
<td>D49C</td>
<td>Read a byte (C08C)</td>
</tr>
<tr>
<td>D4A1</td>
<td>Strip off last two bits XXXXXXX0</td>
</tr>
<tr>
<td>D4A3</td>
<td>Reinitialize offset</td>
</tr>
<tr>
<td>D4A5</td>
<td>Translate byte and merge in (D100)</td>
</tr>
<tr>
<td>D4A8</td>
<td>bits from Auxiliary buffer (D254)</td>
</tr>
<tr>
<td>D4AA</td>
<td>Store byte in Primary buffer (top third) (1000)</td>
</tr>
<tr>
<td>D4B1</td>
<td>Read a byte (C08C)</td>
</tr>
<tr>
<td>D4B5</td>
<td>Increment offset - done yet?</td>
</tr>
<tr>
<td>D4B7</td>
<td>No, then do another &gt;&gt;D4A5</td>
</tr>
<tr>
<td>D4B9</td>
<td>Strip off last two bits XXXXXXX0</td>
</tr>
<tr>
<td>D4BB</td>
<td>Is checksum valid? (D100)</td>
</tr>
<tr>
<td>D4BE</td>
<td>No, then exit with error &gt;&gt;D4CC</td>
</tr>
<tr>
<td>D4C0</td>
<td>Get slot number</td>
</tr>
<tr>
<td>D4C2</td>
<td>Read data register (C08C)</td>
</tr>
<tr>
<td>D4C5</td>
<td>Loop until data valid &gt;&gt;D4C2</td>
</tr>
<tr>
<td>D4C7</td>
<td>Is it last trailing mark ($DE)?</td>
</tr>
</tbody>
</table>

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

**NEXT OBJECT ADDR: D4CA**

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>D4CA</td>
<td>Yes, then continue with carry clear &gt;&gt;D4CD</td>
</tr>
<tr>
<td>D4CC</td>
<td>Set Carry flag indicating error</td>
</tr>
<tr>
<td>D4CD</td>
<td>Get byte we stored away, we have time now</td>
</tr>
<tr>
<td>D4CE</td>
<td>Set proper offset</td>
</tr>
<tr>
<td>D4D0</td>
<td>Store byte in Primary buffer (offset $55)</td>
</tr>
<tr>
<td>D4D2</td>
<td>Return to caller</td>
</tr>
</tbody>
</table>

### 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 (C08C) |
| D4E1 | Delay 25 cycles <D4F0> |
| D4E6 | Has data register changed? (C08C) |
| D4E9 | Yes, then exit >>D4F0 |
| D4EB | Just in case indicate No Device Connected Error |
| D4ED | Decrement count - 256 tries yet? |
| D4EE | No, try again >>D4DE |
| D4F0 | Return to caller |

### D4F1

*CONVERT SLOT/DRIVE TO TABLE OFFSET*

| D4F1 | Preserve A-register |
| D4F2 | Get Unit number DSSS0000 |
| D4F4 | Divide by 16 00000000 |
| D4F8 | Put Drive into Carry 00000000 D |
| D4FA | Strip out Drive 00000000 D |
| D4FC | Roll left 00000000D |
| 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"? (C08E) |
| D507 | No, then continue on >>D50C |
| D509 | Go to error routine >>D50F |
| D50C | Put transition byte from secondary buffer (D300) |
| D50F | into zero page for timing |
| D511 | Use $FF for "sync" byte |
| D513 | Write first "sync" byte (C08F) |
| D519 | Set counter for four more |
| D51C | Delay so that writes occur |
| D51D | Exactly on 40 cycle loops |
D54E Get: "disk byte" (C08D)
D551 Increment index - Done with Auxiliary buffer?
D552 Examine another byte >>D53A
D553 Get the byte of Auxiliary buffer
D554 Put: it into Primary buffer
D555 Lookup five-or with next data byte (1000)
D556 Get: out last two bits XXXXXX00
D557 Get: in X-reg for table lookup
D558 Get: "disk byte" in table (D023)
D559 Increment byte
D560 No, let's do another >>D581
D561 Did: byte byte (Primary buffer) (1000)
D562 Yes, let's test the index, end of this page?
D563 Do we continue on >>D553
D564 You have start on page boundary?
D565 Check if we're done >>D599
D566 Look up five-or with next data byte (1100)
D567 Get: out last two bits XXXXXX00
D568 Get: result in X-reg for table lookup
D569 Get: a "disk byte" in table (D203)
D570 Increment byte
D571 Yes, second condition byte
D572 Exclusive or cycles for correct timing
D573 Setting offset, buffer end or odd byte?
D574 Put: if we're done then >>D599
D575 Lookup five-or with next data byte (1100)
D576 Get: out last two bits XXXXXX00
D577 Get: result in X-reg for table lookup
D578 Get: a "disk byte" in table (D203)
D579 Increment byte
D580 Exclusive or cycles for correct timing
D581 Stri... byte buffer - Put result in carry out last two bits XXXXXX00
D582 9 cycles before write
D583 7 cycles before write
D584 A-register in data register (C08D)
D585 Write data register (C08C) to caller
D586 Puck:...

*** WRITE A BYTE SUBROUTINE  

*** PREPARE BLOCK ROUTINE  

buffer pointer
$2 to buffer address
access top third of buffer >>D58A
result in code below (D630)
direct $14 from buffer address
access middle third of buffer >>D606
result in code below (D625)
direct $9A from buffer address
access bottom third of buffer >>D612
result in code below (D61B)
initialize offset
data byte (bottom third) XXXXXX00 (1000)
in X-reg for table lookup

0030000AB
<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>D628</td>
<td>Use lookup to reposition bits 0000BA00 (D1E0)</td>
</tr>
<tr>
<td>D623</td>
<td>Save result on stack</td>
</tr>
<tr>
<td>D624</td>
<td>Get data byte (middle third) XXXXXXXX (1056)</td>
</tr>
<tr>
<td>D627</td>
<td>Get last two bits 000000CD</td>
</tr>
<tr>
<td>D629</td>
<td>Put in X-reg for table lookup</td>
</tr>
<tr>
<td>D62A</td>
<td>Get current value from stack 0000BA00</td>
</tr>
<tr>
<td>D62B</td>
<td>Merge in new bits using table 00DCBA00 (D1C0)</td>
</tr>
<tr>
<td>D62E</td>
<td>Save result on stack</td>
</tr>
<tr>
<td>D62F</td>
<td>Get data byte (top third) XXXXXXXX (10AC)</td>
</tr>
<tr>
<td>D632</td>
<td>Get last two bits 000000EF</td>
</tr>
<tr>
<td>D634</td>
<td>Put in X-reg for table lookup</td>
</tr>
<tr>
<td>D635</td>
<td>Get current value from stack 01DCBA00</td>
</tr>
<tr>
<td>D636</td>
<td>Merge in new bits using table 01DCBA00 (D1A0)</td>
</tr>
<tr>
<td>D639</td>
<td>Save result on stack</td>
</tr>
<tr>
<td>D63A</td>
<td>Get offset into primary buffer</td>
</tr>
<tr>
<td>D63B</td>
<td>Compute offset into Auxiliary buffer</td>
</tr>
<tr>
<td>D63C</td>
<td>Put data byte just created PEDCBA00</td>
</tr>
<tr>
<td>D63D</td>
<td>Get in Auxiliary buffer (D300)</td>
</tr>
<tr>
<td>D642</td>
<td>Increment offset primary buffer, done yet?</td>
</tr>
<tr>
<td>D643</td>
<td>NO, then do another &gt;&gt;D61A</td>
</tr>
<tr>
<td>D645</td>
<td>Get low order byte of buffer</td>
</tr>
<tr>
<td>D647</td>
<td>Subtract 1 (offset to last byte in buffer)</td>
</tr>
<tr>
<td>D648</td>
<td>Save it for later</td>
</tr>
<tr>
<td>D64A</td>
<td>Get low order byte of buffer</td>
</tr>
<tr>
<td>D64C</td>
<td>Modify code in Write Data Routine (offset) D552</td>
</tr>
<tr>
<td>D64F</td>
<td>Buffer on page boundary? - Yes, skip ahead &gt;&gt;D65F</td>
</tr>
<tr>
<td>D651</td>
<td>Else, compute offset to last byte</td>
</tr>
<tr>
<td>D653</td>
<td>Before page boundary</td>
</tr>
<tr>
<td>D654</td>
<td>Get byte (page boundary -1)</td>
</tr>
<tr>
<td>D656</td>
<td>Point at next byte (page boundary)</td>
</tr>
<tr>
<td>D657</td>
<td>Exclusive-or them together XXXXXXXX</td>
</tr>
<tr>
<td>D659</td>
<td>Strip off last two bits XXXXXXX0</td>
</tr>
<tr>
<td>D65B</td>
<td>Put in X-reg for table lookup</td>
</tr>
<tr>
<td>D65C</td>
<td>Get &quot;disk byte&quot; from table (transition byte) (D203)</td>
</tr>
<tr>
<td>D65F</td>
<td>Save result (0 indicates page boundary)</td>
</tr>
<tr>
<td>D661</td>
<td>Buffer on page boundary? - Yes skip ahead &gt;&gt;D66F</td>
</tr>
<tr>
<td>D663</td>
<td>Get offset to last byte in buffer</td>
</tr>
<tr>
<td>D665</td>
<td>Carry indicates odd or even buffer start</td>
</tr>
<tr>
<td>D666</td>
<td>Get byte (page boundary)</td>
</tr>
<tr>
<td>D668</td>
<td>Did buffer start on odd byte? - Yes skip &gt;&gt;D66D</td>
</tr>
<tr>
<td>D66A</td>
<td>Point at next byte (page boundary +1)</td>
</tr>
<tr>
<td>D66B</td>
<td>Exclusive-or them together</td>
</tr>
<tr>
<td>D66D</td>
<td>Save result</td>
</tr>
<tr>
<td>D66F</td>
<td>Point at last byte in buffer</td>
</tr>
<tr>
<td>D671</td>
<td>Get last byte in buffer XXXXXXX</td>
</tr>
<tr>
<td>D673</td>
<td>Strip off last two bits XXXXXXX0</td>
</tr>
<tr>
<td>D675</td>
<td>Save result (&quot;checksum byte&quot;)</td>
</tr>
<tr>
<td>D677</td>
<td>Get high order byte of buffer</td>
</tr>
<tr>
<td>D679</td>
<td>Modify code in Write Data Routine (D555)</td>
</tr>
<tr>
<td>D680</td>
<td>Get slot number for this operation</td>
</tr>
</tbody>
</table>

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

**NEXT OBJECT ADDR: D620**

---

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>D681</td>
<td>Modify code in Write Data Routine (D55D)</td>
</tr>
<tr>
<td>D69A</td>
<td>Return to caller</td>
</tr>
<tr>
<td>D69B</td>
<td>DETERMINE IF SLOT/DRIVE HAS CHANGED ***************</td>
</tr>
<tr>
<td>D69D</td>
<td>Compare unit number with &quot;current&quot; unit number (D359)</td>
</tr>
<tr>
<td>D69E</td>
<td>Put &quot;current&quot; drive in Carry</td>
</tr>
<tr>
<td>D69F</td>
<td>Has slot changed? - No, then exit &gt;&gt;D6BD</td>
</tr>
<tr>
<td>D6A9</td>
<td>Get &quot;current&quot; slot</td>
</tr>
<tr>
<td>D6AB</td>
<td>Put in X-register</td>
</tr>
<tr>
<td>D6AC</td>
<td>Exit if Slot 0 &gt;&gt;D6BD</td>
</tr>
<tr>
<td>D6AE</td>
<td>Is &quot;current&quot; motor is on? &lt;D4C&gt;</td>
</tr>
<tr>
<td>D6B1</td>
<td>NO, then exit &gt;&gt;D6BD</td>
</tr>
<tr>
<td>D6BB</td>
<td>Wait until &quot;current&quot; motor is off (D378)</td>
</tr>
<tr>
<td>D6BB</td>
<td>Or else timeout &gt;&gt;D6A6</td>
</tr>
<tr>
<td>D6BD</td>
<td>Return to caller</td>
</tr>
<tr>
<td>D6BE</td>
<td>CLEAR IN PHASES *******</td>
</tr>
<tr>
<td>D6CE</td>
<td>Get unit number</td>
</tr>
<tr>
<td>D6C0</td>
<td>Strip drive bit</td>
</tr>
<tr>
<td>D6C2</td>
<td>Put slot 16 in X-Register</td>
</tr>
<tr>
<td>D6C3</td>
<td>Clear phases in case there is (C080)</td>
</tr>
<tr>
<td>D6C6</td>
<td>one of them new-fangled storage (C082)</td>
</tr>
<tr>
<td>D6C9</td>
<td>devices sharing this slot (C084)</td>
</tr>
<tr>
<td>D6CC</td>
<td>Modify code with my (trusty old Disk II. (C086)</td>
</tr>
<tr>
<td>D6CF</td>
<td>Return to caller</td>
</tr>
<tr>
<td>D6D0</td>
<td>CHECK CALLING PARAMETERS *******</td>
</tr>
<tr>
<td>D6D0</td>
<td>Note: For 48-track drives, change byte at $D6E3 from $18 to $40</td>
</tr>
<tr>
<td>D6D8</td>
<td>Check command code</td>
</tr>
<tr>
<td>D6D9</td>
<td>Is it greater or equal to 4?</td>
</tr>
<tr>
<td>D6D4</td>
<td>Yes, indicate error &gt;&gt;D666</td>
</tr>
<tr>
<td>D6D6</td>
<td>Get Block Number</td>
</tr>
<tr>
<td>D6DA</td>
<td>Is Block Number good? (D356)</td>
</tr>
<tr>
<td>D6DD</td>
<td>Yes, if less than $100 &gt;&gt;D668</td>
</tr>
<tr>
<td>D6DE</td>
<td>No, if greater than or equal to $200 &gt;&gt;D666</td>
</tr>
<tr>
<td>D6EA</td>
<td>No, if greater than or equal to $118 &gt;&gt;D668</td>
</tr>
<tr>
<td>D6ED</td>
<td>Indicate error</td>
</tr>
<tr>
<td>D6E7</td>
<td>Return to caller</td>
</tr>
<tr>
<td>D6EB</td>
<td>All is well</td>
</tr>
<tr>
<td>D6E9</td>
<td>Return to caller</td>
</tr>
<tr>
<td>D6EA</td>
<td>$D6EA-$D6FF NOT USED *******</td>
</tr>
<tr>
<td>D6EA</td>
<td>Not used</td>
</tr>
<tr>
<td>ADDR</td>
<td>DESCRIPTION/CONTENTS</td>
</tr>
<tr>
<td>------</td>
<td>----------------------</td>
</tr>
<tr>
<td></td>
<td>The Disk II Device Driver for Version 1.3 changes only in one routine—the &quot;clear phases&quot; subroutine. Phases are now cleared with a &quot;LDA&quot; instead of a &quot;STA&quot; to eliminate bus fights that potentially cause unwanted writing to the 5.25&quot; disk.</td>
</tr>
</tbody>
</table>

**D6BE**

**CLEAR IWM PHASES**

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

**D6D0**

**CHECK CALLING PARAMETERS**

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

- Check command code
- Is it greater or equal to 4?
- Yes, indicate error $D6E6
- Get Block Number
- Is Block Number good? ($356)
- Yes, if less than $1w0 $D6E8
- No, if greater than or equal to $200 $D6E0
- No, if greater than or equal to $118 $D6E8
- Indicate error
- Return to caller
- All is well
- Return to caller

**D6EA**

**$D6EA-$D6FF NOT USED**

- Not used
<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>FF9B</td>
<td><strong>MODULE STARTING ADDRESS</strong></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>* IRQ Handler</td>
</tr>
<tr>
<td></td>
<td>* Resides at $FF9B. Put</td>
</tr>
<tr>
<td></td>
<td>* there by ProDOS Relocator.</td>
</tr>
<tr>
<td></td>
<td>* VERSION 1.2 -- 6 SEP 86</td>
</tr>
<tr>
<td></td>
<td>* VERSION 1.3 -- 2 DEC 86</td>
</tr>
<tr>
<td></td>
<td>* (The IRQ Handler is still the</td>
</tr>
<tr>
<td></td>
<td>* same as it was in Version 1.0.1)</td>
</tr>
<tr>
<td>BF56</td>
<td>Temporary storage 1</td>
</tr>
<tr>
<td>BF57</td>
<td>Temporary storage 2</td>
</tr>
<tr>
<td>BF68</td>
<td>An register savearea</td>
</tr>
<tr>
<td>BF8D</td>
<td>Bank ID byte</td>
</tr>
<tr>
<td>BFD3</td>
<td>IRQ exit code</td>
</tr>
</tbody>
</table>

**GLOBAL PAGE EQUATES**

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>D000</td>
<td>RAM/ROM test byte</td>
</tr>
<tr>
<td>C002</td>
<td>ROM Select</td>
</tr>
<tr>
<td>C08B</td>
<td>BANK1 Select</td>
</tr>
</tbody>
</table>

**EXTERNAL EQUATES**

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>FF9B</td>
<td><strong>IRQ CODE</strong></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Put A-Register on stack</td>
</tr>
<tr>
<td></td>
<td>Get Accumulator value from $45</td>
</tr>
<tr>
<td></td>
<td>and save it (BF56)</td>
</tr>
<tr>
<td></td>
<td>Replace $45 with A-Register</td>
</tr>
<tr>
<td></td>
<td>since it may have been destroyed</td>
</tr>
<tr>
<td></td>
<td>Load Status register</td>
</tr>
<tr>
<td></td>
<td>Restore onto stack</td>
</tr>
<tr>
<td></td>
<td>Isolate B flag - Was it a BRK?</td>
</tr>
<tr>
<td></td>
<td>Yes, skip Interrupt stuff &gt;&gt;FFC2</td>
</tr>
<tr>
<td></td>
<td>Else, Check location $D000 (D000)</td>
</tr>
<tr>
<td></td>
<td>Do we have RAM active</td>
</tr>
<tr>
<td></td>
<td>Yes, indicate so &gt;&gt;FBB3</td>
</tr>
<tr>
<td></td>
<td>Else, indicate ROM</td>
</tr>
<tr>
<td></td>
<td>Update Bank ID byte (BF8D)</td>
</tr>
<tr>
<td></td>
<td>Also save temporarily (BF57)</td>
</tr>
<tr>
<td></td>
<td>Push ($BF50) address of</td>
</tr>
<tr>
<td></td>
<td>routine to bank in Ram and</td>
</tr>
<tr>
<td></td>
<td>call IRQ on the stack</td>
</tr>
<tr>
<td></td>
<td>Push a new P-Register on stack with</td>
</tr>
<tr>
<td></td>
<td>the Interrupt Disable flag set</td>
</tr>
<tr>
<td></td>
<td>Push ($FA41) address less 1 of</td>
</tr>
</tbody>
</table>

**VARIABLES**

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>FFC0</td>
<td>Monitor IRQ on the stack</td>
</tr>
<tr>
<td>FFC8</td>
<td>Select ROM - execution continues in ROM (C002)</td>
</tr>
</tbody>
</table>

**RESET CODE**

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>FFC8</td>
<td>Push ($FA61) address less 1 of (FFD7)</td>
</tr>
<tr>
<td>FFC9</td>
<td>Hardware Reset routine on to stack</td>
</tr>
<tr>
<td>FFD3</td>
<td>Exit via select ROM code above &gt;&gt;FFC8</td>
</tr>
<tr>
<td>FFD6</td>
<td>Address (-1) of Hardware Reset routine</td>
</tr>
</tbody>
</table>

**IRQ CODE**

Called via $BF50 in System Global Page

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>FFD8</td>
<td>Save Accumulator in Global page (BF88)</td>
</tr>
<tr>
<td>FFD9</td>
<td>Restore $45 with original value (BF56)</td>
</tr>
<tr>
<td>FFE0</td>
<td>Select RAM (read &amp; write) (C08B)</td>
</tr>
<tr>
<td>FFE3</td>
<td>use BANK1 (C08B)</td>
</tr>
<tr>
<td>FFE6</td>
<td>Get Bank ID byte (BF57)</td>
</tr>
<tr>
<td>FFE9</td>
<td>Leave via Global Page IRQ exit code &gt;&gt;BFD3</td>
</tr>
</tbody>
</table>

**SFFEC-SFFP9 UNUSED**

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>FFE0</td>
<td>These unused bytes are at $4FEC-$4FP9 when</td>
</tr>
<tr>
<td></td>
<td>FFD9 loaded as part of the &quot;PRODOS&quot; file</td>
</tr>
</tbody>
</table>

**VECTORS**

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>FFFA</td>
<td>NMI Vector</td>
</tr>
<tr>
<td>FFFC</td>
<td>Reset Vector</td>
</tr>
<tr>
<td>FFEE</td>
<td>IRQ Vector</td>
</tr>
</tbody>
</table>
ThunderClock Code -- V1.2 -- 6 SEP 86

ADD DESCRIPTION/CONTENTS

D742 MODULE STARTING ADDRESS

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

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

003A Binary month (1=JAN, 2=FEB, etc.)
003B Binary day of week (0=Sunday, 1=Monday, etc.)
003C Binary day of the month (0-31)
003D Binary hour of the day (0-23)
003E Binary minute of the hour (0-59)

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

0200 Input Buffer
BF0 Global page year-month-day
BF9 Global page hours-minutes

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

D742 Get slot ROM high byte (D750)
D745 Get a screen hole byte for this slot ($038)
D748 and save it on the stack
The two JSR addresses that follow will be
modified by ProDOS Relocator so that they
will access the correct slot ROM.

D74B Write an $A3 to the clock (consult your Thunderclock manual) <$10B>
D74E Read the clock: <$10B>
Reading the clock results in an ASCII string
being placed in the input buffer. A sample
string might be "37,06,04,22,46,57", which is
July (month 07) Saturday (day-of-week 6)
the 4th (day of month 4) 10 PM (hour 22)
46 minutes and 57 seconds after the hour.

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

D751 ---
D752 Five values to convert (ProDOS ignores seconds)
D754 Y-reg is index into
D756 the input buffer ($200)
D759 Strip ASCII from ten's digit
D75D by 10
D762 Add in one's digit ($201)
D766 and subtract off ASCII.
D768 then store as binary in 2-page
D76A Skip over comma
D76B and two digits
D76E More values to convert >>D76

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

D770 Save month in Y-reg
D771 three low Bits of month
D774 three high bits of accum,
D775 combine w/4th day of month,
D777 and store in low byte of DATE (MMDDDDD) (BF9)
D77A Save carry (high bit of month)
D77B Add day of the month
D77D to table value to get Julian date (D7AB)
D780 Is it September? >>D784
D782 Yes, add 3 and carry (see notes below)
D784 Compute Julian date MOD 7
D785
D788 Subtract day-of-week to get year index
D78D Index positive? >>D791
D78F No, make it positive
D791 Index to Y-Reg
D792 Get year from year table (D788)
D795 Get high bit of month in carry
D796 roll it into accumulator
D797 to get high byte of DATE (YYYYMM) (BF9)
D799 Put hour
D79C In high byte of TIME (BF93)
D79F Put minutes
D7A1 in low byte of TIME (BF92)
D7A0 Restore #4ed screen hole value ($038)
D7AB RETURN

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

D7AC January
D7AD February
D7AE March
D7AF April
D7B0 May
ThunderClock Code -- V1.2
DESCRIPTION/CONTENT

D781 June
D782 July
If month>7, value
less than Julian,
is added along wit:
D783 August
D784 September
Note: For Julian,
is three mo,
so that it
D785 October
D786 November
D787 December
6 SEP 86
NEXT OBJECT ADDR: D781

D788 ***** YEAR TABLE *****
This table is good

D788 1990
D789 1989
D78A 1988 {March to Dec}
D78B 1988 {January and
D78C 1987
D78D 1986
D78E 1991

|D78E| 255, the table value |
|    | more than the low byte should be, |
|    | will properly divide by 7. |

**************************************************************************
for the years 1986-91
D742 ** ** GLOB/CONTENTS

BF90 ProDOS DAT
BF92 ProDOS TIM

D742 ******** IIGS ********

C068 One byte \n D742 ******** CLO ** le IIGS, then this
D744 Get IIGS S.
D747 s \n D74A Make sure
D750 Get out of
D751 16-bit mem.
D756 Push 4, m1.
D75A Tool = \n D761 Jump to th D763 Get pre-gp
D766 and restore.
D769 Throw away SOFT SWITCH ********
D76F Store base.
D772 Get year.
D773 and store \n D776 Get day w D777 in range by and index operations.
D778 and save \n D77A Get month \n D77C in range we're in bank 0.
D77D and shift \n D782 Combine vi \n D785 and store

** IMPS **

D742 MODULE START

*********

Beneath Apple ProDOS

IIGS Clock Code ---
ADDR DESCRIPTION
-------------
D742 MODULE START
********

* IIGS CLO
* If PR

IIGS Clock Code --- V1.2 -- 6 SEP 86 NEXT OBJECT ADDR: D742
ADD    DESCRIPTION/CONTENTS
D788 Roll high bit of month into year to put (BF91)
Yyyyyyy in Global Page.
D78B Threw away day of week.
D78C and null byte.
D78E Back to emulation mode.
D78F RETURN
D790 Saved STATEREG Byte.
D791 Vanity code.
D7A1 $D7A1 to $D7BE not used, set to 0.
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 -- Vi.1 -- 18 JUN 84

-------------

2000       MODULE STARTING ADDRESS

******************************************************************************
*                                                                         *
*   PRODOS BASIC INTERPRETER RELOCATOR                                       *
*   LOADED AS THE FIRST TWO BLOCKS                                          *
*   OF BASIC SYSTEM AT $2000.                                               *
*   THIS ROUTINE MOVES THE BASIC                                           *
*   INTERPRETER TO $9A60-$BCCF.                                            *
*                                                                         *
*   BASIC VERSION 1.1 -- 18 JUN 84                                         *
*                                                                         *
*   DISTRIBUTED WITH PRODOS 6 VERSIONS                                     *
*   1.1.1, 1.2, and 1.3.                                                   *
******************************************************************************

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

0000  "PROM" POINTER FOR COPY
0001  "TO" POINTER FOR COPY
0036  CSWL VECTOR
0038  KSWL VECTOR
006F  APPLESOFT START OF STRINGS
0073  APPLESOFT HIMEM
00F2  APPLESOFT TRACE FLAG

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

0200  PATHNAME BUFFER
0260  PREFIX BUFFER
0261  START OF PREFIX NAME
0300  WARMSTART VECTOR
032D  COLDSTART VECTOR
03F0  BRK HANDLER ADDRESS
03F1  RESET HANDLER ADDRESS
03F3  POWER-UP BYTE
03F5  APPLESOFT & VECTOR
03F9  CTL-Y VECTOR

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

0400  FIRST SCREEN BUFFER LINE
0480  SCREEN BUFFER LINE
0628  SCREEN BUFFER LINE

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

BC7A  BASIC INTERPRETER VERSION NUMBER
BE00  BASIC INTERPRETER ENTRY POINT
BE03  8I COMMAND SCANNER (SYNTAX)
BE10  COUNT VECTORS FOR EACH SLOT
BE20  KSWL VECTORS FOR EACH SLOT
BE3C  DEFAULT SLOT NO.
BE3D  DEFAULT DRIVE NO.
BEF3  HIMEM

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

BF00  MACHINE LANGUAGE INTERFACE ENTRY
BF30  LAST DEVICE USED
BF58  MEMORY MAP
BF5B  MACHINE TYPE FLAGS
BF99  SLOTS WHICH CONTAINS CARDS WITH ROM
BF9A  IF 6, NO PREFIX ACTIVE
BFFD  INTERPRETER VERSION NUMBER

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

B800  APPLESOFT ENTRY POINT
FA59  BRK HANDLER
FB2F  INIT SCREEN, MONITOR, ETC.
FC58  CLEAR SCREEN, HOME CURSOR
FEDD  STANDARD CHARACTER OUT
FDF8  CHARACTER OUTPUT TO SCREEN
FE84  SET NORMAL CHARACTER ATTRIBUTE

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

2000  JUMP OVER STARTUP FILENAME >>2047
2006  STARTUP FILENAME LENGTH (7)
2007  'STARTUP'
200E  ALLOW FOR 64 CHAR FILENAME
2047  $00 --> $2400
2048  $02 --> $9A00
2055  COPY 35 PAGES
2058  COPY INTERP TO HIGH MEMORY AT $9A00 <20C4>
205D  PAGE FOLLOWING INTERP IMAGE IS...
205F  BASIC GLOBAL PAGE IMAGE
2061  COPY THAT TO $8800 <20C4>
2064  TO GET 40-COL DISPLAY, SEND A CTRL-U
20B1 ********** ERROR EXIT ****************************

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

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

20C7 TO $2/3
20CA A PAGE AT A TIME >>20C4
20D0 COUNT PAGES
20D3 RETURN

20D4 ************* CSWL INTERCEPT / CONTINUE **********************

20D5 "$J" APPLESOFT PROMPT?
20D6 NO...DON'T PRINT WHATEVER IT IS >>20D3
20D8 YES, APPLESOFT DONE SETTING UP (BE10)
20DB POINT CSWL TO STANDARD OUTPUT
20DE CHECK LAST DEVICE USED (BF30)
20DF ISO WIRE TO STANDARD OUTPUT (2238)
20EB DRIVE ONE OR TWO? >>20EE
20EE STORE DEFAULT DRIVE (D) (BE30)
20F2 ISOLATE SLOT FROM DEVICE NO.
20FA AND STORE DEFAULT SLOT (A) (BE3C)
20FA GET SLOT BYTE SHOWING CARDS PRESENT? (BF99)
2102 PICK OFF ITS BITS ONE BY ONE
2108 SET OUTVCS AND INVCVS TO $5C60 (BE10)
210B FOR ALL SLOTS WITH ROMS IN THEM (BE20)

BI Relocator -- V1.1 -- 18 JUN 84

NEXT OBJECT ADDR: 2066

2115 ---
211B SET HIMEM TO $9600
211D IN VARIOUS PLACES
2124 GET A DEFAULT PREFIX? (BF9A)
2127 NO >>214E
2129 YES, AND GET PREFIX <BF98>
212F ERROR? >>215B
2136 READD PREFIX FOR "/'S (0280)
213B AND COUNT THEM IN $223E (223E)
213E ---
213F FOR A COUNT OF SUBLEVELS >>2136
2146 MORE THAN JUST VOLUME NAME? >>216F
214B NO, MLI: SET PREFIX <BF00>
214E MLI: ONLINE <BF00>
2154 ERROR? >>215B
2156 SET VOL NAME LENGTH (0281)
215B NONE THERE? >>215B
215F ADD ONE TO NAME LENGTH (0280)
2164 AND PREFIX IT WITH A "/" (0281)
2167 MLI: SET PREFIX <BF00>
216D ERROR? >>215B

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

216F MLI: GET FILE INFO <BF00>
2172 FIND "STARTUP" FILE
2175 ERROR? >>215B
2186 SAVE LENGTH OF STARTUP FILE NAME (2236)
217D COPY NAME TO $200 (205E)
2186 FIRST COMMAND WILL BE "STARTUP"
218B CHECK NUMBER OF SUBLEVELS (223E)
219D MORE THAN JUST VOL? >>219D
2192 MLI: SET PREFIX <BF00>
2198 ANY STARTUP FILE NAME? (2236)
219B YES, SKIP MESSAGE >>21C1
219D SET TRUE KSWL <2209>
21A2 PRINT 'PRODOS BASIC 1.1' (2267)
21A0 PRINT 'COPYRIGHT ...' (2283)
21B6 SKIP THREE LINES

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

21C1 ---
21C3 COPY WARNSTART JMP TO PAGE 3 (21FF)
21C9 AND COLDSTART (U3D3)
21CC AND CLI-Y (03F8)
21C7 PRINT "PRODOS" (2206)
21D2 TO $BE0J (CMD SCANNER) (03F5)
21D6 COPY BRK HANDLER JMP ALSO (2202)
21E7 AND RESET JMP (83F2)
21F2 SET POWER-UP BYTE ACCORDINGLY (03F4)
BI Relocator -- $ Supplement

1.1 -- 18 JUN 84

NEXT OBJECT ADDR: 21F7

21F7 SET APPL

ON/CONTENTS

21F9 GET

21FC PUT IT, AI

21FF GO TO INIT

SOFT IN NON-TRACE MODE

PREPERS VERSION NUMBER, (BC7A)

SYSTEM GLOBAL PAGE, (3FFD)

202 ERPRETER >>BE80

2202 BREAK HAND

2204 RESER HAND VECTOR ADDRESSES ************

2206 APPLICATION

2209 ***********

ADDRESS FOR PAGE 3

2209 GET KSWL

2213 RETURN OF FIRST KSWL INTERCEPT ************

2217 FOLLOWED

2219 RETURN TO CURRENT DEVICE HANDLER (BE20)

2219 OF FIRST COMMAND (2086)

221A ************

221A CSWL (2086)

221C KSWL (2286)

221E GET FILE (1) INTERCEPT ADDR

221F FILENAME (9) INTERCEPT ADDR

2221 15 BITES

2230 THIS BYTE INFO PARMLIST

2231 SET PRES RESERVED FOR OTHER GET_FILE PARMS (NOT USED)

2232 FOR PRES NOT USED

2234 NULL PRES PARMLIST

2235 AT $2234

2236 SAVED LEX

2237 ONLINE 9A

2239 PUT VOLUME

223B SET PREFIX

223C PREFIX IS NAME AT $281

223E NUMBER OF X PARMLIST

223F UNA

2267 SUBLEVELS IN PREFIX +1

2283 ILE TO EXECUTE BASIC SYSTEM ***

PRODOS BASIC 1.1

COPYRIGHT APPLE, 1983-84
BASIC Interpreter (BI) -- V1.1 -- 18 JUN 84  NEXT OBJECT ADDR: 9A00

ADDR  DESCRIPTION/CONTENTS

9A00  MODULE STARTING ADDRESS

******************************************************************************
* PRODOS BASIC INTERPRETER (BI)                                           *
* THIS CODE STARTS IN THE THIRD Block of the file BASIC.SYS.                *
* IT PERFORMS COMMAND HANDLING                                             *
* FOR ALL BUILT-IN PRODOS COMMANDS AND SUPPORTS BASIC'S FILE HANDLING.     *
******************************************************************************

VERSION 1.1 -- 18 JUN 84

* DISTRIBUTED WITH PRODOS VERSIONS
  1.1.1, 1.2, AND 1.3.

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

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

0024  CURSOR HORIZONTAL
0028  SCREEN LINE BASE ADDR
0029  MONITOR PROMPT CHARACTER
0036  CRT DISPLAY VECTOR (CSWL)
0037  KEYBOARD INPUT VECTOR (KSWL)
0039  SCRATCH POINTER AND LOOP COUNTER
003A  SCRATCH POINTER AND LOOP COUNTER
003D  POINTER TO APPLESOF VARIABLES
003F  APPLESOF: LINE NUMBER
0050  APPLESOF: START OF PROGRAM PTR
0068  APPLESOF: LOMEM (START OF VARS)
006A  APPLESOF: START OF ARRAY VARS PTR
006E  APPLESOF: START OF FREE AREA PTR
0070  APPLESOF: START OF STRINGS PTR
0073  APPLESOF: HIMEM (END OF STRINGS)
0074  APPLESOF: CURRENT LINE BEING EXECUTED
0075  APPLESOF: WIDTH OF CURRENT LINE

0080  APPLESOF: ADDR OF LINE AFTER FINDLINE
009C  APPLESOF: END OF PROGRAM PTR
00A0  APPLESOF: START OF PROGRAM PTR
00B9  APPLESOF: PROGRAM LOCKED (PROTECTED)
00D8  APPLESOF: OKERR ACTIVE FLAG
00DE  APPLESOF: OKERR CODE
00F2  APPLESOF: TRACE ACTIVE FLAG
00FB  APPLESOF: INTERNAL STACK

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

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

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

AE06  EXTERNAL COMMAND ENTRY TO BI
AEB6  PRINT ERROR MESSAGE ENTRY TO BI
AEBF  PRODOS ERROR CODE
AE10  OUTPUT VECTORS FOR ALL SLOTS
AE19  CURRENT OUTPUT VECTOR
AE22  CURRENT INPUT VECTOR
AE34  PRODOS INTERCEPT VECTORS (INPUT/OUTPUT)
AE38  BI'S INTERNAL REDIRECTION VECTORS
AE3C  DEFAULT SLOT
AE3D  DEFAULT URIVE
AE3E  A REGISTER SAVE AREA
AE3F  X REGISTER SAVE AREA
AE40  Y REGISTER SAVE AREA
AE41  TRACE FLAG (APPLESOF TRACE ON/OFF)
AE42  IMMEDIATE COMMANDS=680, DEFERRED=1
AE43  EXEÇ FILE ACTIVE=$80
AE44  READ FILE ACTIVE=$80
AE45  WRITE FILE ACTIVE=$80
AE46  READ/WRITING PREFIX ACTIVE=$30
AE47  DIRECTORY FILE BEING ACCESSSED
AE49  FREE STRING SPACE DURING GARBAGE COLLECT
AE4A  BUFFERED I/O BYTE COUNT
AE4B  INDEX INTO INPUT COMMAND LINE
AE4C  LAST OUTPUT CHAR TO PREVENT RECURSION
AE4D  NUMBER OF OPEN NON-EXEC FILES
AE4E  EXEC FILE BEING CLOSED FLAG
AE4F  READ FILE IS TRANSLATED DIRECTORY
AE50  VECTOR TO EXTERNAL COMMAND HANDLER
AE52  LENGTH-1 OF EXTERNAL COMMAND STRING
AE53  COMMAND NUMBER
AE54  PARAMETERS ALLOWED FOR THIS COMMAND
BASIC Interpreter (BI) -- V1.1 -- 18 JUN 84  NEXT OBJECT ADDR: 9A03

(SEE BIT DEFINITIONS IN TABLE LATER)

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

BE58 A KEYWORD VALUE
BE5A B KEYWORD VALUE
BE5D E KEYWORD VALUE
BE5F L KEYWORD VALUE
BE61 S KEYWORD VALUE
BE62 D KEYWORD VALUE
BE63 F KEYWORD VALUE
BE65 R KEYWORD VALUE
BE68 @ KEYWORD VALUE
BE6A T KEYWORD VALUE
BE6B SLOT NUMBER FROM IN# OR PR#
BE70 ISSUE MIL CALL AND XLAKE ERROR CODES
MIL PARAMETER FIELDS

BEA3 CREATE: ACCESS CODE
BEA4 CREATE: FILE ID
BEA5 CREATE: AUX ID
BEA7 CREATE: FILE KIND
BE4 SET/GET FILE INFO: Parm Count
BE7 SET/GET FILE INFO: Access Code
BE8 SET/GET FILE INFO: File ID
BE9 SET/GET FILE INFO: AUX ID
BEB SET/GET FILE INFO: File Kind
BEC SET/GET FILE INFO: Blocks Used
BEF SET/GET FILE INFO: Modify Date/Time
BEC ONLINE/GET/SET MARK/EOF/BUF: REF NUM
BEG ONLINE/GET/SET MARK/EOF/BUF: MARK/BUF
BFC OPEN: SYSTEM BUFFER
BED OPEN: REF NUM RETURNED
BE2 NEWLINE: REF NUM
BE3 NEWLINE: NEW LINE CHAR (ALWAYS CR)
BE6 READ/WRITE: REF NUM
BE7 READ/WRITE: DATA ADDRESS
BE9 READ/WRITE: LENGTH OF DATA
BEH READ/WRITE: ACTUAL LENGTH TRANSMITTED
BE9 OPEN/FLUSH: REF NUM
BEF BASIC HIMEM VALUE

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

BF03 QUIT VECTOR
BF30 LAST DEVICE USED
BF56 MEMORY UTILIZATION BIT MAP
BF94 OPEN FILE LEVEL
BF9A PREFIX ACTIVE FLAG (IF NONZERO)
BASIC Interpreter (BI) -- V1.1 -- 18 JUN 84  NEXT OBJECT ADDR: 9A2F

---

9AA3 ********** SET CSWL/KSWL INTERCEPTS **************

9AA3 ---
9AA4 COPY VDOSIO VECTORS (BE34)
9AA7 TO CSWL
9AB1 AND KSWL
9AB9 EXIT TO CALLER

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

(IMMEDIATE MODE)

9ABA IS EXEC FILE ACTIVE? (BE43)
9ABF YES, SAVE REGISTERS <9F62>
9AC2 AND GO READ EXEC FILE FOR INPUT COMMANDS >>9AF5
9AC5 NO EXEC FILE, RESTORE REAL CSWL/KSWL <9AB0>
9AC8 NO, READ A KEY FROM KEYBOARD <FD10>
9ACB RETURN?
9ACD NO, EXIT >>9AEB
9ACP YES, SAVE REGISTERS <9F62>
9AD2 STORE IT IN LINE BUFFER (9200) --> THIS ENTRY CALLED BY EXEC TO PROCESS A COMMAND STRING STORED AT $20
9AD5 GO PROCESS THE COMMAND STRING (A677)
9AD8 CHECK COMMAND NUMBER RETURNED FROM PARSE (BE53)
9ADB EXIT BI RIGHT NOW? >>9AE2
9ADD NO, COMMAND RETURNED WITH ERROR CODE? >>9AF0
9ADF NO, RESTORE Y REG (BE48)
9AE2 RETURN A BACKSPACE TO CALLER OF KEYBOARD
9AE4 AND A LINE INDEX OF 2E0
9AEB EXIT THE BI >>9AEB
9AE8 RESTORE CALLER'S REGISTERS <9F6C>
9AEB AND EXIT BI BY INSTALLING INTERCEPTS >>9A3D

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

9AEE ERROR=3, "NO DEVICE CONNECTED"
9AP0 MAIN ENTRY: STORE ERROR CODE (BE0F)
9AP3 AND IN APPLESOFT OERR
9AP5 CHECK BI STATE (BE42)
9AP8 MEMORIZE WHETHER IT'S IMMEDIATE MODE
9APD SET A HIGH FILE LEVEL FOR NON-EXEC FILES (B994)
9BB2 NO ACTIVE READ/WRITE FILES OR PREFIX READ (BE44)
9BB8 CLOSE ALL OPEN FILES AT OR ABOVE (BEDE)
9BEE FILE LEVEL = $0F
9B10 MLI: CLOSE (ALL) <BE70>
9B13 ERROR? >>9B27
9B15 WRITE ANY DATA I HAVE BUFFERED <A000>

---
BASIC Interpreter (BI) -- V1.1 -- 18 JUN 84

DESCRIPTION/CONTENTS

9B84 ERROR? >>9B27
9B85 PUT FILE LEVEL BACK TO ZERO
9B86 NOW FLUSH ALL OPEN FILES
9B87 MLI: FLUSH (ALL) <<BE70>
9B88 ---
9B89 ASSUME MODE WILL BE 4 (DEFERRED)
9B8A MEMORIZE WHETHER BASIC ON ERROR ACTIVE
9B8B DEFERRED MODE CURRENTLY? >>9B30
9B8C NO, STILL IMMEDIATE MODE (MODE=0)
9B8D ---
9B8E SET MODE AS DEFINED ABOVE <9F76>
9B8F RESTORE BI'S CSWL/KSWL INTERCEPTS <9AA3>
9B90 GET ERROR CODE (BE0F)
9B91 BASIC ONERR ACTIVE? THEN GO HANDLE IT >>9B4D
9B92 NO, JUST PRINT ERROR MESSAGE <<8B5C
9B93 CLOSE EXEC FILE IF ONE IS OPEN <<B25B
9B94 DEFERRED MODE? >>953
9B95 IMMED. MODE, PRINT RETURN AND... <<9FAB
9B96 WARMSTART APPLESOFT >>D43F
9B97 RESTORE STACK FOR BASIC
9B98 PASS ERROR CODE TO BASIC
9B99 ---
9B9A JUMP INTO APPLESOFT ERROR HANDLER >>D865
9B9B ********** RETURN TO IMMED. MODE ***********************************
9B9C CLEAR APPLESOFT ERRNUM
9B9D WILL LOOK FOR "#" FROM APPLESOFT
9B9E SET NORMAL VIDEO IN APPLESOFT <P273>
9B9F RESTORE TRUE CSWL/KSWL <9A00>
9BA0 TRY TO WRITE BUFFERED DATA <<9F4D
9BA1 RESET MODE/SET UP BI'S INTERCEPTS <9A17>
9BA2 RESTORE REGISTERS <9F6C>
9BA3 GO TO PROCESS IMMED. INPUT REQUEST >>9ABA
9BA4 ********** INPUT INTERCEPT: MODE=4 OR 8 ******************************
9BA5 SAVE REGISTERS <<9F62>
9BA6 PREFIX INPUT ACTIVE? (BE46)
9BA7 NO >>9B7E
9BA8 YES, GO DO SPECIAL HANDLING >>9D67
9BA9 ELSE, IS READ FILE ACTIVE? <<BE44>
9BAA NO >>9B86
9BAA YES, GO DO SPECIAL HANDLING FOR THAT >>9C16
9BAC ELSE, IS EXEC FILE ACTIVE? <<BE43)
9BAD NO >>9B9F
9BAE YES, GET PROMPT CHARACTER
9BAF IT BETTER NOT BE A "[
9BB0 IT IS, RETURN TO IMMEDIATE MODE >>9B58
9BB1 ELSE, SET TRUE CSWL/KSWL <9A00>
9B9A AND PASS CALLER'S AXREG TO REMOVE CURSOR (BE3E)
9B9B RESTORE Y-REGISTER (BE40)
9B9C REMOVE CURSOR AND GET A KEYPRESS <<FD10>
9B9D BACKSPACE?
9B9E NO, EXIT BI >>9BAC
9B9F YES, CHECK PROMPT
9BA0 IF ITS A "...
9BA1 THEN EXIT WITH THE BACKSPACE >>9BAA
9BA2 ELSE, IF AT START OF LINE, REPROMPT >>9B94
9BA3 MIDDLE OF LINE, RETURN A BACKSPACE
9BA4 EXIT BI TO CALLER >>9ABD
9B9F ********** READ EXEC FILE *********************************************
9B9G REMOVE CURSOR FROM SCREEN
9B9H CHECK PROMPT CHARACTER
9B9I IF ITS A ""
9B9J DO THINGS DIFFERENTLY >>9BF2
9B9K CHECK KEYBOARD (C000)
9B9L NO KEY READY? >>9BCE
9B9M GOT A KEY, IS IT CONTROL-C?
9B9N NO, IGNORE IT >>9BBD
9B9O YES, CLOSE EXEC FILE <B2FB>
9B9P IMMEDIATE MODE? <<BE42)
9B9Q NO >>9C01
9B9R YES, CLEAR KEYBOARD STORED (C61)
9B9S AND GO START NEW LINE >>9C01
9B9T SET UP FOR EXEC LINE READ <<9D68
9B9U READ A LINE TO $200 <<9C6C
9B9V ERROR? >>9BFA
9B9W SAVE REGISTERS <<9F62>
9B9X HOP INTO LOOP >>9BDE
9B9Y ---
9B9Z BACKSCANING $200 BUFFER ($200)
9B9C FORCING THE MSB ON
9B9D RESTORE TRUE CSWL/KSWL <9A00>
9B9E GO PROCESS COMMAND LINE <9AD5>
9B9F CHECK COMMAND NUMBER (BE33)
9B9G IMMEDIATE EXIT? IF NOT, GET NEXT LINE >>9B9C
9B9H RETURN

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

9B9F GET SET TO READ EXEC LINE <<9D68>
9B9F5 READ SINGLE CHARACTER PER CALL <<9C40>
9B9F8 NO ERRORS, EXIT TO CALLER NOW >>9BF1
Beneath Apple ProDOS 8k

BASIC ERROR RECOVERY ************

BASIC Interpreter (E)

--- 9E <B245> -- SD OF DATA? 9C53
            EXIT THEN >>9C13

--- JUST STOP EXECING

         9BFA CLOS EXEC
         9BF0 WAS ERROR "CHARACTER UNDER CURSOR"
         9BF1 NO, REAL ERROR >>9B3B
         9C01 ELSE, OK >>9AF0
         9C03 GET CURSOR
          9C05 IF IN MIDDLE
            >>9F2F FILE ACTIVE ***********
          9C07 ELSE, CHANGE
          9C08 AND RETURN
          9C0D RETURN
          9C0E GET SCREEN
             IMMEDIATE MODE >>9B58
          9C10 AND EXIT THE CURSOR FROM SCREEN (BE3E)
          9C13 REAL ERROR, SJ (C086)
             >>9C31

--- 9C16 *********** IF IT CONTROL-C?
          9C16 GET PROMPT >>9C31
          9C18 IF ITS A "J"
          9C1C THEN RESET
          9C1F ELSE, REMOVE CURSOR
          9C24 CHECK KEYBO
             DERY FILE? (BE47)
          9C27 NO KEYBO
          9C29 GOT A KEY, OR = "?"
          9C2B NO, IGNORE >>9C42
          9C2D CLEAR STROBE
             ENTIRE LINE >>9C67
          9C30 RETURN

          9C31 GET PROMPT
          9C33 IS THIS A D
             O BYTE FROM INPUT FILE >>9C34
          9C36 YES >>9C49
          9C38 NO, IS I/O PROM? >>9C3A
          9C3C ELSE, READ A
          9C3F ERROR? >>9C
          9C41 RETURN
             >>9C42 READ SINGLE
             DEREAd ONE BYTE (BED9)
          9C45 ERROR? >>9C570
          9C47 RETURN
          9C48 AGC TO MAXIMUM AGAIN (BE5F)

--- 9C48 SAVE CURRENT
          9C4A IN L KEYWORD
          9C50 SET UP TO N
          9C55 MLI: READ <
          9C58 ERROR? >>9C
          9C5A PUT COUNT B
          9C60 GET FIRST C

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

--- 9C67 *********** READ NEXT LINE OF FILE ***********
          9C67 REMOVE CURSOR FROM SCREEN (BE3E)
          9C6C ---
          9C6E MLI: READ <BE76>
          9C71 ERROR? >>9C66
          9C73 GET LENGTH ACTUALLY TRANSMITTED (BED8)
          9C76 NOTHING? >>9C8E
          9C79 GOT SOMETHING, FIND END OF DATA (BED7)
          9C7D FETCH LAST BYTE OF LINE (BE1F)
          9C82 IS IT A RETURN CHARACTER?
          9C84 NO, LEAVE LINE ALONE >>9C8E
          9C86 YES, WAS L KEYWORD GIVEN? (BE57)
          9C88 YES, LEAVE IT BE >>9C8E
          9C8D ELSE, CHOP OFF THE RETURN ITSELF
          9C8E AND EXIT WITH A RETURN
          9C90 RESTORING Y REG AS YOU GO (BE40)
          9C94 RETURN

--- 9C95 *********** READING DIR FILE ***********
          9C95 "." PROMPT?
          9C97 YES, EXIT RIGHT NOW >>9C8E
          9C99 ELSE, REMOVE CURSOR FROM SCREEN (BE3E)
          9C9E SET 80 COLUMNS
          9CAB MLI: GET MARK <BE76>
          9CAB ERROR? >>9D1F
          9C9E ARE WE AT BEGINNING OF THIS FILE? (BEC8)
          9CBE NO, CONTINUE >>9CDF
          9C82 YES, CAT FLAG = 2
          9CB7 READ DIRECTORY HEADER <B15D>
          9CBA ERROR? >>9D1F
          9CBC REF NUM TIMES 32 (BED6)
          9CC7 SET THE L VALUE OF THIS DIR FILE IN (BCFF)
          9CCA THE OPEN FILE LIST TO THE ENTRY LENGTH (B8C8)
          9CCD AND THE NUMBER OF ENTRIES PER BLOCK (BED8)

--- *********** FORMAT DIRECTORY NAME ***********
          9CDE G0 FORMAT NAME OF DIRECTORY <BBB8>
          9CD3 STORE THE LENGTH OF LINE AT $200
          9CD8 PUT A RETURN CHAR AT END OF LINE
          9CDE AND EXIT TO CALLER
          9CDE RETURN
BASIC Interpreter (B1) -- V1.1 -- 18 JUN 84 NEXT OBJECT ADDR: 9CDE

ADDR DESCRIPTION/CONTENTS

9CDE GET CAT FLAG (BE4F)
9CF2 IF ZERO, GO PROCESS INDIVIDUAL ENTRIES >>9D22
9CBE IF MINUS, GO DO SUMMARY LINE OR EXIT >>9CF9
9CBE POSITIVE, ASSUME NULL LINE WANTED
9CBE DROP CAT FLAG BY ONE (BE4F)
9CBE IF ZERO, JUST GO PRINT A BLANK LINE >>9CD3

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

9CDE ELSE, BLANK OUT $200 AND <A6C>
9CF2 UNPACK "NAME TYPE BLOCKS ETC..." <9F80>
9CF5 LINE LENGTH IS 80
9CF7 GO RETURN IT TO CALLER >>9CD3

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

9CF9 DO SUMMARY LINE?
9CFB NO, JUST EXIT (ALL DONE) >>9D1C
9CFD YES, DROP CAT FLAG SO EXIT NEXT TIME (BE4F)
9D00 CLEAR READ/WRITE COUNT (BED9)
9D0A ALL: READ <BE46>
9D0D FORMAT BLOCKS FREE AND INUSE SUMMARY LINE <B0E7>
9D11 GET REF NUM (BED6)
9D14 AND COPY TO GET/SET LIST (BEC7)
9D1B NO ERRORS, EXIT >>9CF5
9D1A ERROR, JUMP TO BI ERROR EXIT >>9D1F
9D1C "END OF DATA" ERROR
9D1F GO TO BI ERROR EXIT >>9APS

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

9D22 SET DIR ENTRY NUM COUNTER TO -1
9D27 GET REF NUM (BED6)
9D2A *32
9D2F USE AS INDEX TO GET ENTRY LENGTH (BCFF)
9D35 AND ENTRIES PER BLOCK FROM OPEN FILE LIST (BD08)
9D3B POSITION ON EVEN BLOCK BOUNDARY (BEC9)
9D41 AND GET SECTOR OFFSET (BEC8)
9D45 SKIP FILE/DIR ENTRIES UNTIL POSITIONED TO (BCBB)
9D48 CURRENT POSITION IN THIS BLOCK (BCB7)
9D50 READ NEXT DIR ENTRY FROM FILE <BD11>
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

BASIC Interpreter (B1) -- V1.1 -- 18 JUN 84 NEXT OBJECT ADDR: 9D5E

ADDR DESCRIPTION/CONTENTS

9D61 FORMAT FILE/DIR ENTRY INTO $201 <A4C4>
9D64 AND RETURN IT TO CALLER >>9CF5
9D67 ******* PREFIX INPUT ACTIVE *******

9D67 PROMPT = "J"?
9D69 NO, ALL IS WELL >>9D6E
9D6B YES, RETURN TO IMMEDIATE MODE NOW >>9B58
9D6E REMOVE CURSOR FROM SCREEN (B3E)
9D75 PREFIX NO LONGER ACTIVE AFTER THIS (BE46)
9D7B COPY PATHNAME BUFFER (PREFIX) (BCBC)
9D7E TO $200 (B1FC)
9D8A RETURN WITH IT TO BASIC (BC3C)
9D89 RETURN

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

9D9A SET READ REF NUM FOR EXEC FILE (B6A3)
9D9C READ TO $200
9D95 FOR $EF BYTES OF LENGTH
9D9A (OR UNTIL A RETURN CHAR)
9D92 RETURN

9D9A ******* OUTPUT INTERCEPT: MODE = C *******

(LOOK FOR CONTROL-D)

9D93 SAVE REGISTERS <9F62>
9D96 PRINTING A CONTROL-D?
9D98 NO >>9DC1
9D9A YES, WRITE OUT ANY BUFFERED DATA <9F44>
9D9D NOTHING IN COMMAND LINE (BE4B)
9DB0 READ FILE INACTIVE (BE44)
9DB3 WRITE FILE INACTIVE (BE45)
9DB6 PREFIX READ INACTIVE (BE46)
9DB8 SET MODE = 8 FROM NOW ON <9F76>
9DBE RESTORE REGS AND EXIT >>9FC6

9DCA GOT A CONTROL-D...
9DC3 SET MODE = 4 FROM NOW ON <9FC6>
9DC6 RESTORE REGISTERS <9FC6>
9DC9 OUTPUT CHARACTER AND EXIT >>B7F1

9DCC ******* OUTPUT INTERCEPT: MODE = 8 *******

(ASSEMBLE COMMAND LINE)

9DCC SAVE REGISTERS <9F62>
9D20 SAVE CHAR IN COMMAND LINE (0200)
9D05 WAS IT A RETURN?
9D07 YES, READY TO ROLL >>9D67
9D09 NO, BUMP CHARACTER COUNTER (BE4B)
Beneath Apple ProDOS Supplement

BASIC Interpreter (BI) -- VI.1 -- 18 JUN 84
Next object addr: 9DC

9DCC AND EXIT TO CALLER >>9DE3
9DE3 OOPS! LINE TOO LONG >>9DF0
9DF0 "SYNTAX ERROR" >>9AF0
9DE3 ELSE, RESTORE X REG AND EXIT (BE3F)
9DE6 RETURN

9DE7 ---
9DE9 NULL LINE? >>9DF6
9DF3 NO, PUT BACK TRUE CSWL/KSWL <9A00>
9DF8 SYNTAX SCAN CMD LINE <A677>
9DF8 ERROR? >>9DF8
9DF8 NO, PUT BACK BI'S INTERCEPTS <9A8D>
9DF6 ---
9DF8 MODE = 4 NOW <9F76>
9DF8 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
9E13 IF NOT, EXIT TO CALLER... (BE40)
9E0E WITH CHARACTER CHANGED TO SCREEN >>9A74

9E11 GET INDEX TO TEMPORARILY BUFFERED CHAR (BE4A)
9E16 STORE INTO BUFFER JUST ABOVE HIMEM
9E18 BUMP INDEX (BE4A)
9E1E OK >>9E22
9E20 BUFFER FULL, SAVE REGISTERS <9F62>
9E22 WRITE BUFFER OUT TO DISK <9FEE>
9E26 ERROR? >>9DE0
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)

9E2C PRINTING A "#"? (9F61)
9E2F NO >>9E49
9E31 YES, SAVE X REGISTER (BE3F)
9E35 RETURN ADDR IS IN APPLESOFT... (0103)
9E38 TRACE ROUTINE...
9E3C AT $D812? (0104)
9E41 YES >>9E56
9E43 NO, RESTORE REGISTERS (9F61)
9E49 IS WRITE FILE ACTIVE? (BE45)
9E4C NOPE >>9E6C
9E4E YES, PRINTING A "#"?
DOES BI KNOW WE ARE TRACING? (BE41)
YES, REAL LIVE TRACE THEN >>9F39
ELSE, PICK UP NEXT TOKEN ON LINE
IS IT A TOKEN? >>9EFC
OR END OF LINE? >>9EEE
NEITHER, DECREMENT STRING SPACE CTR (BE49)
OK >>9EEC
COMPUTE SIZE OF FREESPACE IN PAGES
AT LEAST 3 PAGES AVAILABLE?
YES >>9EE5
NO, WRITE BUFFERED DATA >>9FF4
AND THEN GARBAGE COLLECT >>A844
COMPUTE FREE SPACE NOW
AND SAVE IN STRING SPACE CTR (BE49)
GET NEXT TOKEN
JUMP BACK INTO APPLESOF TO EXECUTE IT >>D820
STORE TOKEN IN PROMPT
LOOK UP TOKEN IN BI'S TOKEN TABLE (B799)
IT'S NOT ONE BI IS INTERESTED IN >>9EEE
IT IS INTERESTING, CHANGE BRANCH (9EFD)
AND JUMP TO ONE OF THE FOLLOWING: >>9EEF
IF OR PRINT: PROMPT = 0
CLEAR OUT LAST CHAR SAVEBASE (BE4C)
GO TO MODE = C NEXT TIME THRU (B603)
BEGIN LOOKING FOR COMMANDS (BE38)
NOW GO PROCESS THE IF OR PRINT >>9F2E
LIST: PROMPT = 1
DON'T LOOK FOR COMMANDS NOW
GO DO IT >>9F2E
CALL: PROMPT = 2
DON'T LOOK FOR COMMANDS NOW
GO DO IT >>9F2E
LET: DECREMENT STRING CTR
AND GO BACK FOR NEXT Token >>9EEC
TRACE: TURN TRACE ON (BE41)
THEN CONTINUE BELOW >>9F2A
NOTRACE: DROP INTO BACKGROUND TRACE (BE41)
CHANGE TOKEN TO "TRACE"
FORCE ON APPLESOF TRACE
---
GO BACK TO APPLESOF TO PERFORM IT >>D820
********** REAL TRACE ACTIVE ************
Beneath Apple ProDOS Supplement

BASIC Interpreter (BI) -- V1.1 -- 18

ADD R DESCRIPTION/CONTENTS

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

9FB0 NOTHING IN BUFFER AT FIRST...
9FB6 GET A NIBBLE FROM PACKED MSG...
9FB9 NON-ZERO, COMMON CHARACTER...
9FBC IF ZERO, GET NEXT NIBBLE <97
9FBE AND CONVERT TO UNCOMMON CHAR.
9FC8 --
9FC1 GET THE LETTER THIS NIBBLE OUTLINE
9FC4 ZERO? THEN END OF MESSAGE... 
9FC5 GET INDEX INTO OUTPUT BUFFER
9FC9 AND STORE THE CHARACTER THERE.
9FCC BUMP INDEX (BE4B)
9FCF AND CONTINUE >>9FB6
9FD1 RETURN

9FD2 ****************** UNPACK MESSAGE BYTES

9FD2 GET NEXT MSG BYTE (BA48)
9FD5 WORKING ON SECOND NIBBLE? >
9FDE NO, TAB INDICATOR? >>9DFB
9FDF NO, ISOLATE HIGH NIBBLE
9FE9 NEXT TIME GET LOW NIBBLE
9FE6 RETURN

9FDE --
9FE6 GET TAB POSITION (BA48)
9FE3 AND BUMP OUTPUT PTR ACCORDI
9FE7 THEN GO BACK FOR NEXT NIBBLE
9FED BUMP BYTE PTR FOR NEXT TIME
9FED ISOLATE LOW NIBBLE
9FEE NEXT TIME GET HIGH NIBBLE
9FED RETURN

9FEE ****************** WRITE ONE BUFFERED MESSAGE

9FEE SET UP COUNT OF 0001
9FF2 AND JUMP INTO ROUTINE BELOW

9FF4 ****************** WRITE BUFFERED DATA

9FF4 WRITE BUFFERED DATA <0000>> >>9FC0
9F79 00? THEN EXIT >>A81C
9FFA ERROR, POP OUT OF THIS SUBROUTINE
9FFD AND GO TO ERROR HANDLER >>9E

REPRESENTS (BA28)
9FD1
9F9E
1F20

A022 FOR A GARBAGE COLLECT WORKAREA (BC70)
A033 IT IS J+1 PAGES IN LENGTH (BC7E)
A038 END OF STRING AREA IS AT END OF FREE AREA (BC86)
A044 GO COLLECT CONSTANT STRINGS NOW <A885>
A043 THEN EXIT

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

(FAST APPLESOFT STRING GARBAGE COLLECTION)

<p>| GENERAL PURPOSE BUFHR |
| TOP OF OLD STRINGS |
| HIMEM --&gt; |
| NEW STRINGS BUILDING |</p>
<table>
<thead>
<tr>
<th>DOWN</th>
</tr>
</thead>
<tbody>
<tr>
<td>----</td>
</tr>
<tr>
<td>V</td>
</tr>
<tr>
<td>------------------</td>
</tr>
<tr>
<td>OLD STRINGS</td>
</tr>
<tr>
<td>----</td>
</tr>
</tbody>
</table>

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.

A0800 ******************** WRITE ALL BUFFERED DATA ***************************

A0800 --
A0802 GET BUFFERED DATA COUNT (BE4A)
A0805 NONE BUFFERED? >>A01B
A0807 STORE BUFFERED DATA COUNT IN RW PARMS (BE9)
A080F ML1: WRITE <BE7B>
A0815 NOTHING BUFFERED NOW, COUNT=0 (BE4A)
A0819 ERROR? >>A01C
A081B NO, EXIT
A081C RETURN

A081D ****************** SPECIAL GARBAGE COLLECT ********************

(PULL OUT STRING CONSTANTS ALSO)

A081D DO GARBAGE COLLECTION NORMALLY FIRST <<A044>
A0823 ERROR? >>A043
A0824 START OF STRING AREA = PROGRAM START PTR (BC84)
A0827 USE GENERAL PURPOSE BUFFER (A00F-HIMEM)
A0F7  *************** COLLECT SIMPLE STRINGS **************
A0F7
A0F8 ADD 7 BYTES TO $3E/$3F PTR FOR NEXT VAR
A0F9 PTR AT ARRAYS NOW?
A0FA IF SO, WE ARE DONE >> A12B
A0FB IS THIS A STRING VARIABLE?
A0FC NO >> A0F7
A0FD MAKE ABSOLUTELY SURE
A0FE GET MSB OF STRING POINTER
A0FF IS IT WITHIN MY RANGE? (BC7F)
A100 NO >> A0F8
A101 NO >> A0F7
A102 YES, PULL IT OUT AND TACK IT TO HIMEM <<
A103 ALL WENT WELL, GET NEXT VARIABLE >> A0F8
A104 IF ERROR, EXIT NOW
A105 NORMAL EXIT TO CALLER
A106 RETURN
A107
A108  *************** COLLECT STRING ARRAYS ***************
A109 FIND THE NEXT ARRAY <A15C>
A10A NO MORE >> A12B
A10B GOT ONE, GET MSB OF ITS STRING PTR
A10C WITHIN MY RANGE? (BC7F)
A10D NO >> A146
A10E NO >> A146
A10F YES, PULL IT OUT AND TACK IT TO HIMEM <<
A110 AND CONTINUE WITH NEXT ARRAY ELEMENT >>
A111 ERROR EXIT
A112
A113  *************** FIND NEXT STRING ARRAY **************
A114 FIND THE NEXT ARRAY
A115
A116  *************** COLLECT STRING ARRAYS ***************
A117
**GENERAL PURPOSE ALLOCATE**

A1F7 STORE THAT (BB47)
A1FA GO GARBAGE COLLECT TO GET SPACE <A044>
A1FD ERROR? >>A24A
A201 HOW MANY FREE PAGES ARE THERE?
A203 ARE THERE ENOUGH? (BB47)
A206 IF NOT, "RAN TOO LARGE" MSG
A20B TOO FEW... >>A24A
A20A GOT ENOUGH? >>A3A--TOP OF FREESPACE
A211 AND $3C--NEW TOP AFTER ALLOCATION
A21B COMPUTE LENGTH OF STRINGS FOR COPY
A229 COPY STRINGS DOWN "N" PAGES IN MEMORY <A35B>
A22F SUBTRACT "N" FROM STRING ADDRESS MSB'S (BB47)
A235 ADJUST ALL POINTERS IN SIMPLE & ARRAY VARS <A39F>
A23A OLD HIMEM BECOMES BUFF ADDR HIGH WATER MARK (BB49)
A241 NEW HIMEM IS "N" PAGES LOWER
A246 FIND PAGE JUST BEYOND A FILE BUFFER (BC6B)
A249 RETURN
A24A --
A24B RETURN

**FREE BUFFER**

A24C GARBAGE COLLECT STRINGS <A044>
A24F ERROR? >>A299
A255 PUT HIMEM-$100 INTO $3A/3B
A259 AND HIMEM+$400 INTO $3C/3D
A25F (COPY LSH'S)
A266 BC92 = LENGTH OF STRINGS (BC92)
A270 COPY STRINGS UP 4 PAGES <A37F>
A275 PREPARE TO ADJUST THEM BY $400 (BC87)
A27B NEW HIMEM+$400
A27D ADJUST ALL STRINGS ADDR UP BY $400 <A39F>
A283 ARE WE FREEING BOTTOM-MOST BUFFER?
A285 YES, DONE! >>A2B3
A28B CHECK OPEN FILE COUNT (BB4D)
A28B DONE OPEN? (HOW CAN THAT BE?) >>A297
A28D WHICH FILE'S BUFFER IS NEXT TO HIMEM?
A292 SEARCH UNTIL IT IS FOUND... >>A29A
A297 --
A299 RETURN IF NO FILE IS USING THIS BUFFER
A29A --
A29B GIVE THAT FILE THE BUFFER PRESSED TO US (BC99)
A29E (SURE HOPE THAT FILE WAS FLUSHED!) (BC93)
A2A9 PASS FILE REP NUM TO MIL (BC7)
A2AE MIL: GET NEW BUFFER <SE70>
A2B1 ERROR? >>A299
A2B3 --
A2B4 RETURN
A2B5 ******************** GETBUFRI: GET A BUFFER ********************

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

A2B5 ALLOCATE A BUFFER OF ANY SIZE (A=PAGES) <A1P7>
A2B8 ERROR? >>A300
A2BD FIND FIRST PAGE OF BUFFER (B84A)
A2C4 GET FILE OPEN COUNT (B84D)
A2C7 IN FILE OPEN? >>A2EA
A2C9 BUMP BUFFER PAGE PTR BY $000 (B849)
A2CD TO POINT TO PREVIOUSLY ALLOCATED
A2CF BUFFER. (B849)
A2D2 FIND OPEN FILE WITH THIS BUFFER (B893)
A2D7 GET IT, (B8C9)
A2DA SET FILE BUFFER REAL LOW IN MEMORY <<A532>
A2DD THEN SET IT TO NEW BUFFER LOCATION <A29B>
A2E0 BELOW ALL OTHERS (B8C9)
A2E7 DO THIS FOR EACH OPEN FILE...
A2EB THEREBY INSERTING A BLANK BUFFER >>A2D2
A2ED IS EXEC FILE ACTIVE? (B843)
A2F0 NO, DONE >>A2FF
A2F2 YES,
A2F4 MOVE EXEC BUFFER DOWN ALSO <A352>
A2FD AND BUMP UP ABOVE IT
A2FF EXIT TO CALLER
A300 RETURN

A301 ******************** FREEBUFRI: FREE BUFFER ********************

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

A301 GET COUNT OF OPEN FILES (B84D)
A305 INDEX THIS BY 4 PAGES PER FILE
A306 ADD TO HIMEM MSB
A308 SAVE THIS AS TOP OF BUFFERS (B849)
A30D THEN SET UP BOTTOM AS HIMEM MSB (B84A)
A310 GET OLD ORIGINAL HIMEM (BEFORE ANY BUFFERS) (B8F8)
A313 SAME AS THIS ONE?
A315 THEN NOTHING ELSE TO DO >>A300
A317 ASSUME NO BUFFERS BY REPLACING OLD HIMEM
A319 ANY EXEC FILE OPEN? (B843)
A31C NO, CONTINUE >>A323
A31E YES, MOVE EXEC BUFFER TO OLD HIMEM <A2F2>
A321 AND GO MOVE HIMEM DOWN BY $000 >>A341
A323 ELSE, START WITH TOP BUFFER (B849)
A326 ANY OPEN FILES? (B84D)
**BASIC Interpreter (BI) -- V1.1 -- 18 JUN 84** NEXT OBJECT ADDR: A39E

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>A39F</td>
<td>*********** ADJUST ALL STRING ADDRS ***********************</td>
</tr>
<tr>
<td></td>
<td>(BC87 HAS ADDITIVE ADJUSTMENT FACTOR)</td>
</tr>
<tr>
<td>A39F</td>
<td>USE LOMEM PAGE AS MSB FOR $3E/3F</td>
</tr>
<tr>
<td>A3A3</td>
<td>GET LOMEM LSB</td>
</tr>
<tr>
<td>A3A5</td>
<td>AND END OF SIMPLE VARS PAGE</td>
</tr>
<tr>
<td>A3AF</td>
<td>JUMP INTO THE LOOP &gt;&gt;A3A5</td>
</tr>
<tr>
<td>A3AA</td>
<td>---</td>
</tr>
<tr>
<td>A3AB</td>
<td>SKIP ONE SIMPLE VARIABLE</td>
</tr>
<tr>
<td>A3AF</td>
<td>---</td>
</tr>
<tr>
<td>A3B1</td>
<td>OVERFLOW? &gt;&gt;A3B5</td>
</tr>
<tr>
<td>A3B3</td>
<td>YES, BUMP MSB</td>
</tr>
<tr>
<td>A3B5</td>
<td>FINISHED WITH SIMPLE VARS?</td>
</tr>
<tr>
<td>A3B9</td>
<td>(CHECK BOTH MSB AND LSB OF PTR)</td>
</tr>
<tr>
<td>A3BB</td>
<td>---</td>
</tr>
<tr>
<td>A3BC</td>
<td>YES... &gt;&gt;A3D2</td>
</tr>
<tr>
<td>A3BE</td>
<td>NO,</td>
</tr>
<tr>
<td>A3C0</td>
<td>LOOK AT A SIMPLE VARIABLE</td>
</tr>
<tr>
<td>A3C5</td>
<td>SKIP INTEGRAL AND REAL VARS &gt;&gt;A3AA</td>
</tr>
<tr>
<td>A3C7</td>
<td>(DOUBLE CHECK MSB)</td>
</tr>
<tr>
<td>A3CB</td>
<td>ITS A STRING, POINT TO ITS LEN/ADDR</td>
</tr>
<tr>
<td>A3CC</td>
<td>ADJUST IT IF NECESSARY &lt;A3F9&gt;</td>
</tr>
<tr>
<td>A3CF</td>
<td>THEN SKIP OVER IT &gt;&gt;A3AA</td>
</tr>
<tr>
<td>A3D2</td>
<td>COPY ARRAYS STARTING LSB</td>
</tr>
<tr>
<td>A3D4</td>
<td>(MSB IS IN X REGISTER NOW) (BC81)</td>
</tr>
<tr>
<td>A3D7</td>
<td>---</td>
</tr>
<tr>
<td>A3DB</td>
<td>FIND A STRING ARRAY &lt;A15C&gt;</td>
</tr>
<tr>
<td>A3DB</td>
<td>NO MORE? THEN DONE... &gt;&gt;A40C</td>
</tr>
<tr>
<td>A3DD</td>
<td>---</td>
</tr>
<tr>
<td>A3E0</td>
<td>ADJUST ITS ADDRESS IF NEEDED &lt;&lt;A3F9&gt;</td>
</tr>
<tr>
<td>A3E6</td>
<td>SKIP TO NEXT STRING ELEMENT OF ARRAY</td>
</tr>
<tr>
<td>A3EE</td>
<td>AT END OF THIS ARRAY YET? (BC81)</td>
</tr>
<tr>
<td>A3F1</td>
<td>NO... &gt;&gt;A3DD</td>
</tr>
<tr>
<td>A3F3</td>
<td>(CHECK MSB ALSO)</td>
</tr>
<tr>
<td>A3F7</td>
<td>YES... GO GET NEXT ARRAY &gt;&gt;A3D7</td>
</tr>
<tr>
<td>A3F9</td>
<td>*********** ADJUST A STRING ADDRESS **********************</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>A3F9</td>
<td>GET STRING LENGTH</td>
</tr>
<tr>
<td>A3FB</td>
<td>IGNORE NULL STRINGS &gt;&gt;A40C</td>
</tr>
<tr>
<td>A3FD</td>
<td>POINT TO MSB OF ADDRESS</td>
</tr>
<tr>
<td>A3FF</td>
<td>IS STRING STORED OUTSIDE OF PROGRAM?</td>
</tr>
<tr>
<td>A4BJ</td>
<td>NO, LEAVE IT ALONE &gt;&gt;A40C</td>
</tr>
<tr>
<td>A4BD</td>
<td>STORE ABOVE LOMEM, ADD FACTOR TO MSB</td>
</tr>
<tr>
<td>A40C</td>
<td>THEN EXIT</td>
</tr>
</tbody>
</table>

**BASIC Interpreter (BI) -- V1.1 -- 18 JUN 84** NEXT OBJECT ADDR: A40C

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>A40D</td>
<td>*********** COMPRESS ALL ASOFT VARS *********************</td>
</tr>
<tr>
<td></td>
<td>THIS ROUTINE SQUEALS ALL APPLESOFT VARS UP AGAINST THE</td>
</tr>
<tr>
<td></td>
<td>BOTTOM OF THE STRINGS</td>
</tr>
<tr>
<td></td>
<td>HIMEM --&gt;</td>
</tr>
<tr>
<td></td>
<td>STRINGS</td>
</tr>
<tr>
<td></td>
<td>ARRAY VARS</td>
</tr>
<tr>
<td></td>
<td>SIMPLE VARS</td>
</tr>
</tbody>
</table>

| A48D | GARBAGE COLLECT FIRST <A81D>                            |
| A410 | ERROR? >>A471                                           |
| A412 | COMPUTE LENGTH OF SIMPLE AND ARRAY VARS                 |
| A417 | AND SAVE IT (BC89)                                      |
| A427 | NEXT, COMPUTE LENGTH OF SIMPLE VARS ONLY                |
| A42B | AND SAVE IT (BC89)                                      |
| A435 | SUBTRACT VAR LENGTH FROM STRING START                   |
| A437 | TO FIND A PLACE TO PUT THE VARS UNDER (BC92)            |
| A446 | THE STRINGS (START ON AN EVEN PAGE BOUND)               |
| A440 | $3C/$3D --> PLACE TO PUT VARS                           |
| A447 | $3A/$3B --> START OF VARS (ROUNDED TO EVEN PAGE         |
| A449 | ALIGNEMENT)                                             |
| A44F | COPY VARS UP AGAINST STRINGS <A37F>                     |
| A454 | STORE START OF VARS PTR (BC88)                          |
| A457 | BUMPING PAGE NUMBER BY ONE                              |
| A463 | SUBTRACT THIS PTR FROM HIMEM TO COMPUTE (BC98)           |
| A466 | TOTAL LENGTH OF COMBINED VARS/STRINGS                   |
| A468 | ALSO, SAVE HIMEM MSB IN CASE THEY ARE MOVED             |
| A471 | DONE, EXIT                                              |

**BASIC Interpreter (BI) -- V1.1 -- 18 JUN 84** NEXT OBJECT ADDR: A472

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>A472</td>
<td>*********** REEXPAND COMPRESSED VARS *********************</td>
</tr>
<tr>
<td></td>
<td>THIS ROUTINE MOVES SIMPLE AND ARRAY VARS</td>
</tr>
<tr>
<td></td>
<td>BACK DOWN TO LOMEM</td>
</tr>
<tr>
<td></td>
<td>HIMEM --&gt;</td>
</tr>
<tr>
<td></td>
<td>STRINGS</td>
</tr>
<tr>
<td></td>
<td>FREE SPACE</td>
</tr>
</tbody>
</table>
A6E3  DOES  >>A73C
A6E7  YES  >>A73C
A6E8  NO  >>A6F6
A6E9  NAME >>A6EA
A6EA  END OF FILENAME (COMATCHING??) >>A6EA
A6E2  NO  >>A6F6
A6F2  YES  >>A6EA
A6F3  "/" >>A6EA
A6F5  NO  >>A761
A6F7  YES  >>A761
A6F9  NO  >>A761
A6FB  NO  >>A761
A6FD  AT OPERAND ON LENGTH (BCBC)
A6F0  FIND KEYWORD AT BEGINNING OF 7F <<A767
A704  PUT NAME ATOP PATHNAME 
A709  SAVE NEXT PATHNAME >>A767
A712  FOUND AT CHAR (OTHER OUT OF PATHNAME
A715  COPY PATH OR RETURN  >>A72C
A718  (XG05) >>A798  >>A72C  (XG05) >>A798
A71E  CHECK NAME EXPECTED >>A73C
A722  NOT CONSIS WELL >>A73C
A724  RETURN  >>A798  >>A73C
A726  NO  >>A798  >>A73C
A72A  YES  >>A72C
A72B  SIG  >>A72C
A72C  NO  >>A72C
A72D  NON ALPHABETIC >>A72C
A72F  IS  >>A72C
A730  NO  >>A72C
A731  56  >>A72C
A732  YES  >>A72C
A733  RETURN  >>A72C
A734  END OF FILENAME >>A72C
A735  FIND KEYWORD >>A72C
A73C  RETURN  >>A72C
A73E  "?" (A ONE BYTE S)
A740  IF >0,  >>A79B >>A79B
A742  ELSE  >>A79B
A745  CONVERT SLOTTED # >>A73C
A74A  FUT I1 >>A761
A75D  FOUND SLOTTED VALUE >>A79C
A762  CONVERT SLOTTED # >>A79C
A765  ERROR  >>A79C
A767  GET CONSOLE ERROR >>A79C
A75A  >B >>A79C
A75C  NO  >>A79C
A75E  YES  >>A79C
A761  RETURN  >>A79C
A76A  SECOND ELSE ON EB079
A76D  NO  >>A761
A76F  YES  >>A76F
A778  NAME  >>A73C
A78B  DO NOT COMA OR RETURN  >>A79B
A796  NO  >>A79B
A798  YES  >>A79B
A79C  REFRESH  >>A79B
A79D  EXIT WHEN >>A761
A79F  YES  >>A761
A7AC  NO  >>A761
A7A4  NULL OUT PATHNAME (BCBC)
A7A5  MARK THAT WE WILL HAVE ONE SOON >>A79B
A7B0  ADD PREFIX TO FILENAME <<A78D>>  >>A79B
A7B4  ERROR >>A78D
A7BC  GET COMMAND NUMBER (BE53)
A7BF  "?" AS INDEX INTO TABLE >>A78D
A802  GET ADDRESS OF COMMAND HANDLING >>A78D
A808  AND STORE IT FOR INDIRECT JUMPS (SUBROUTINE (B8E9)
A80F  (BCAC)
BASIC Interpreter (B1) -- V1.1 -- 18 JUN 84
-------------------------------

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 <B700>
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 **********************
A83B   LOAD BI CODE FOR "SYNTAX ERROR"
A83D   AND RETURN WITH ERROR CONDITION
A83C   RETURN
A83D   ********** ADD PREFIX TO PATHNAMES ***********
A83D   GET SLOT NUMBER (BE61)
A844   PUT SLOT IN HIGH 3 BITS
A846   ADD DRIVE TO TOP BIT AND SHIFT SLOT DOWN (BE62)
A84E   ...TO FORM THE UNIT NUMBER (BE67)
A853   READ THE PATHNAME PREFIX TO $201 (BE68)
A85D   MLI: ONLINE <BE70>
A860   ERROR? >>A83B
A865   DEFAULT DRIVE = Parsed DRIVE (BE3D)
A86B   DEFAULT SLOT = Parsed SLOT (BE3C)
A871   PATHNAME1 STARTS WITH "/"?
A873   THEN ITS ALREADY GOT A PREFIX >>A866
A878   ELSE, GET LENGTH OF PATHNAME
A87A   BUMP IT BY 2 (TO ALLOW FOR "/")
A882   WITH PREFIX WILL IT EXCEED 64 CHARS?
A887   YES, "SYNTAX ERROR" >>A877
A889   NO, UPDATE LENGTH TO INCLUDE PREFIX (BCBC)
A88F   ---
A893   AND COPY PATHNAME1 FORWARD TO MAKE NOON (BCBD)
A89C   Put a "/" AT THE BEGINNING
A8A1   AND THE END (BCBD)
A8A4   COPY PREFIX JUST READ TO START OF PATHNAME1 (0200)
A8AA   GET COMMAND NUMBER (BE53)
A8AD   "OPEN"?
A8AF   YES, DONE NOW! >>A8E6
A8B1   "APPEND"?
A8B3   YES, DONE NOW! >>A8E6

A8B5   *** EXEC"?
A8B7   YES, DONE NOW! >>A8E6
A8B9   NO, GET LENGTH OF PATHNAME2 (0200)
A8BE   COMBINE THIS WITH PREFIX LENGTH (0201)
A8C1   MORE THAN 64 CHARS?
A8C6   IF SO, "SYNTAX ERROR" >>A8E7
A8C8   ---
A8CF   COPY PATHNAME2 FORWARD TO MAKE ROOM (0281)
A8DB   PUT A "/" IN FIRST
A8DD   THEN THE PREFIX AND ANOTHER SLASH (0281)
A8E9   ---
A8E7   DONE!
A8EB   ********** KEYWORD LOOKUP **********************
A8B8   ZERO THE ACCUMULATOR <A837>
A8BE   NINE POSSIBLE KEYWORDS IN TABLE
A8ED   COMPARE AGAINST EACH (BE6B)
A8F0   FOUND IT? >>A927
A8F5   NO, IS IT "T"? (FILE TYPE)
A8F7   YES, OK THEN >>A8FC
A8F9   ELSE, BAD KEYWORD >>A839
A8FC   IT'S "T", IS IT PERMITTED ON THIS CMD?
A901   NO, ERROR >>A923
A906   ELSE, MARK WE HAVE "T" (BE56)
A90B   START WITH TYPE INDEX OF 0 (BCAD)
A910   INDICATE WHERE T VALUE IS TO GO (BCBE)
A913   AND GO PARSE ONE CHAR <AA3A>
A916   NOTHING THERE??? >>A8F9
A918   IS IT A ?
A91A   YES, HE GAVE TYPE IN HEX >>A976
A91C   IS IT ALPHABETIC?
A91E   NO, CONVERT DECIMAL TYPE >>A960
A920   ELSE, GO LOOKUP TYPE NAME IN TABLE >>A9B6
A923   ---
A924   "INVALID PARAMETER"
A926   RETURN
A927   GET BIT POSITION OF THIS KEYWORD (B975)
A92A   IGNORE "/" >>A947
A92C   IS THIS KEYWORD PERMITTED? (BE55)
A92F   NO, NOT WITH THIS COMMAND ANYWAY >>A923
A931   S OR D?
A933   NO >>A941
A935   YES, ALREADY FOUND IT ON THIS LINE? (BE57)
A939   YES, DON'T CHANGE DRIVE DEFAULT >>A947
A93A   ELSE, ASSUME DRIVE = 1
A941   MARK WE HAVE SLOT/DRIVE (BE57)
A947   GET SIZE-1 IN BYTES OF VALUE (B97F)
BASIC Interpreter (BI) -- V1.1 --18 JUN 84    NEXT OBJECT ADDR: A954

ADDR  DESCRIPTION/CONTENTS

A954    AND OFFSET TO VALUE IN STORAGE AREA (B0AE)
A957    FLUSH TO NON-BLANK <AA3A>
A959A   NOTHING ELSE THERE >>A9B0
A959C   IS NEXT CHAR A "$"?
A959E   YES, GO CONVERT HEX - ELSE, FALL THRU >>A976

A960    ********** CONVERT DECIMAL NUMBER **********
A960J   SAVE LINE INDEX (BE48)
A963    CONVERT/ADD ONE DECIMAL DIGIT TO ACCUM <AA5C>
A966    OK... >>A96C
A968    OVERFLOW? THEN "RANGE ERROR" >>A983
A969A   BAD DIGIT? THEN "SYNTAX ERROR" >>A9B0
A969C   RESTORE LINE INDEX (BE48)
A969F   FLUSH TO NEXT NON-BLANK <AA3A>
A972    AND GO BACK TO CONVERT NEXT DIGIT >>A960
A974    ALL DONE, END OF LINE OR COMMA >>A98F

A976    ********** CONVERT HEX NUMBER **********
A976J   FLUSH TO NEXT NON-BLANK (SKIP "$") <AA3A>
A979    NOTHING LEFT? >>A9B0
A97B    SAVE LINE INDEX (BE4B)
A97E    CONVERT HEX DIGIT <AAAE>
A981    OK... >>A987
A983    OVERFLOW? THEN "RANGE ERROR" >>A983
A985    BAD DIGIT? THEN "SYNTAX ERROR" >>A9B0
A987    RESTORE LINE INDEX (BE4B)
A98A    FLUSH TO NEXT NON-BLANK <AA3A>
A98D    AND GO TO CONVERT NEXT DIGIT >>A97B

A98F    ********** STORE KEYWORD VALUE **********
A98FJ   HOW MANY BYTES TO CHECK?
A994    ALL HAVE BEEN CHECKED? >>A999E
A996    NO, INSURE MSB'S OF ACCUM ARE ZERO (BCAF)
A999    IF NUMBER IS A SHORT INTEGER >>A9B3
A9A1    COPY ACCUM TO PROPER PARM STORAGE CELL (BCAF)
A9A8    RESTORE LINE INDEX (BE4B)
A9AF    AND EXIT

A9B0    "SYNTAX ERROR" JUMP >>A839
A9B3    "RANGE ERROR" JUMP >>A75E

A9B6    ********** STORE KEYWORD VALUE **********
A9B6    ----
A9B8    COPY J CHARACTER TYPE TO ACCUM (BCAF)
A9BE    (COPIED ALL J?) >>A9C7
A9C0    (GET NEXT CHAR IGNORING BLANKS) <AA3A>
A9C5    MUST HAVE 3 CHARACTERS! >>A9B0

BASIC Interpreter (BI) -- V1.1 --18 JUN 84    NEXT OBJECT ADDR: A9C7

ADDR  DESCRIPTION/CONTENTS

A9C7    SAVE LINE INDEX (BE4B)
A9CA    INITIALIZE NAME INDEX TO ZERO
A9CF    HAVE ALL 13 BEEN CHECKED?
A9D1    YES, NO MATCH >>A9B0
A9D4    ELSE, INDEX*3 (BCAD)
A9DB    COMPARE TYPE GIVEN (BCAF)
A9DB    TO TYPES IN TABLE (B977)
A9DE    (IGNORE MSB'S)
A9DF    NO MATCH ALREADY... >>A9E9
A9E3    ELSE,
A9E5    CHECK ALL THREE CHAR'S >>A9D8
A9E7    THEY ALL MATCH WE FOUND IT >>A9EE
A9E9    NOT THE RIGHT ONE, (BCAU)
A9EC    GO TRY THE NEXT ONE >>A9CA
A9EE    REVERSE NAME INDEX
A9F5    AND GET TYPE VALUE FROM TABLE (B989)
A9F8    STORE IT IN TYPE VALUE STORAGE AREA (BE6A)
A9FF    RESTORE LINE INDEX (BE4B)
A9F9    AND EXIT

AA00    ********** COPY PATHNAME2 **********
AA00    GET NEXT CHARACTER <AA4A>
AA03    AND STORE IT INDEXED OFF $280 (B280)
AA07    COMMA?
AA09    YES, DONE >>A37
AA0B    BLANK?
AA0D    YES, DONE >>A37
AA0F    RETURN?
AA11    YES, OUT NOW >>AA48
AA13    PATHNAME TOO LONG? (BCCA)
AA16    NO, CONTINUE COPYING >>AA00
AA18    ELSE, SET NOT-EQUAL CONDITION
AA1A    AND EXIT

AA1B    ********** COPY COMMAND NAME INTO TXTBUF **********
AA1B    SET INDICES
AA1F    GET NEXT NON-BLANK <AA4A>
AA22    COPY TO TXTBUF (BCBD)
AA26    COMMA?
AA2B    YES, DONE >>A37
AA2A    BLANK?
AA2C    YES, DONE >>A37
AA2E    RETURN?
AA30    YES, DONE >>AA48
AA32    AT MAX LENGTH (8)? (BCCA)
AA35    NO, CONTINUE >>AA1F
AA37    ELSE, SET NOT-EQUAL CONDITION
AA39    AND EXIT
BASIC Interpreter (B1) -- V1.1 -- 18 JUN 84  NEXT OBJECT ADDR: AA39

---

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 >>AA68
AA64 NOT NUMERIC, EXIT WITH CARRY SET
AA65 AND Z-FLAG RESET
AA67 RETURN
AA68 ISOLATE DECIMAL PORTION OF DIGIT
AA6B CURRENT VALUE OF ACCUM... (BCB1)
AA6E >1,703,936?
AA70 YES, OVERFLOW >>AA94
AA74 PUSH ENTIRE ACCUM ONTO STACK (BCAF)
AA7B ACCUM*2 (ROL IT ONCE) <AAD7>
AA7E ACCUM*4 (AND AGAIN) <AAD7>
AA81 ---
AA85 ACCUM*4+ACCUM -- ACCUM*5 (BCAF)
AA91 FINALLY, ACCUM*5+2 -- ACCUM*10 <AAD7>
AA94 ---
AA95 ACCUM OVERFLOW? >>AA9A
AA97 NO, ADD NEW DIGIT TO ACCUM (BCAF)
AA99 AND STORE IT (BCAF)
AA9D NO CARRY? >>AAAD
AA90 GOT CARRY, PROPAGATE IT THRU ACCUM (BCB0)
AAA0 OVERFLOW ERROR
AAA1 NORMAL EXIT

---

BASIC Interpreter (B1) -- V1.1 -- 18 JUN 84  NEXT OBJECT ADDR: AA39

---

AAAE ************ CONVERT HEX DIGIT AND ADD *********************

AAAE NUMERIC?
AAB0 NO >>AAAB
AAAB YES >>AAC4
AAAC NON-NUMERIC, HOW BOUT "A" THRU
AAABD "F"?
AAABE YES >>AAC2
AAABF ---
AAAF NO, GET OUT NOW
AAAC1 RETURN
AAAC2 "A" THRU "F", CONVERT TO $BA-$BF
AAAC4 ISOLATE DIGIT
AAACB SHIFT ACCUM 4 BITS LEFT TO MAKE ROOM <AABD>
AAACB (WATCH OUT FOR OVERFLOW) >>AAA
AAAD9 OR IN NEW NIBBLE (BCAF)
AAAD3 AND REPLACE IN ACCUM LSB (BCAF)
AAAD6 DONE

---

AAAD7 ************ SHIFT 3 BYTE ACCUM LEFT A BIT *************

AAAD7 SHIFT THE THREE BYTE WORK ACCUM (BCAF)
AAAE0 RETURN

---

AAAE1 ************ SCAN CMD TABLE FOR COMMAND *************

AAAE1 START WITH LAST COMMAND IN TABLE
AAAB6 IS IT A ~ COMMAND? (BCB0)
AAABE NOPE >>AAAF
AAABD YES, SPECIAL COMMAND NUMBER (BE53)
AAAF0 ZERO LENGTH COMMAND STRING (BE52)
AAAF3 CONTINUE >>AB12
AAAF5 FIRST COMMANDS IN TABLE ARE 8 CHARS
AAAF7 GET INDEX TO NEXT NAME (BE53)
AAAFD SAME LENGTH AS LAST NAME? >>AB05
AAAFP NO,
AAAB02 NAMES ARE ONE BYTE SHORTER FROM NOW ON (BE52)
AB05 ---
AB06 COMPARE HIS NAME TO MY TABLE (BCBD)
AB0C NOT IT... >>AB25
AB10 COMPARE ENTIRE NAME >>AB06
AB12 FOUND IT? GET COMMAND INDEX (BE53)
AB15 *2 FOR MOST THINGS
AB17 PICK UP PERMITTED PARMS BITS (B92A)
AB23 EXIT HAPPLY
AB24 RETURN
BASIC Interpreter (B1) -- V1.1 -- 18 JUN 84
NEXT OBJECT ADDR: ABB0

ADDR  DESCRIPTION/CONTENTS

ABB2  ********** "RUN" COMMAND
ABB2  NO INPUT FILE ACTIVITY
ABB7  NO APPLESOFT ERROR SHIELD
ABB8  GOT PATHNAME YET?
ABBD  NO, ERROR >>ABB5
ABBF  YES, LOAD PROGRAM >>ABB6
ABBC  ERROR? >>AC14
ABCF  NO, CLEAR VARIABLES >>FE
ABB7  CLEAR ERROR FLAG >>D665
ABBC  POSITION TO LIMS >>AC97
ABB8  RESTORE MY INTERESTS >>E776
ABB2  CLEAR COMMAND NUMBER IF ER IF GIVEN >>AC97
ABB2  JUMP INTO APPLESOFT SUCH AS, MODE = 4 >>D7D2
ABB5  ********** CLEAR COMMANDS, NO RUN PROGRAM >>D7D2
ABB0  SET NORMAL (NON-LOAD) NUMBER ETC.
ABDA  SEARCH CHARACTER FOR
ABDD  NO COMMAND NUMBER YET >>AC97
ABE8  NO PROMPT IF TRACER IS "#" (9F61)
ABEB  SET MODE=4 <<EXECUTE (BE53)
ABE9  "SYNTAX ERROR" IF IN
ABEC  ********** "LOAD" COMMAND
ABB0  NGS GO WRONG >>AB39
ABED  LOAD PROGRAM <<SYS> AND
ABE0  ERROR? IF NOT, FAIL
ABF1  ********** WARMSTART >>HRU TO WARMSTART >>AC14
ABF2  CLEAR APPLESOFT, SET WARMSTART BI
ABF4  RESET MODS/SET INTS
ABF9  CURSOR HOME >>CST POINTERS >>D665
ABFB  GO WARMSTART APPLESOFT SET POINTERS <<SA17>
ABFE  ********** LOAD A PROFILE >>D43F
ABFE  CLOSE ALL OPEN FILES >>AM
AC01  ERROR? >>AC14
AC03  GO LOAD FILE <<AC15
AC06  ERROR? >>AC14
AC00  SET LOMEM = ARRAYS
AC0A  ALL TO END OF PROGRAM
AC14  RETURN

AC2F  DO NOT FREESTART LOADED
AC15 ********** READ A PROGRAM FROM A FILE **********

AC15 READ REQUESTED
AC17 TYPE = BAS ASSUMED
AC19 OPEN THE FILE <B194>
AC1C ERROR? >>AC14
AC20 MLI; GET EOF <BE70>
AC23 ERROR? >>AC14
AC27 APPLESOFT PROGRAM START --> READ DATA (BED7)
AC2A ADD TO THAT THE EOF MARK TO ... (BEC8)
AC2D SET AD PARM --> END OF PROGRAM IMAGE (BE58)
AC3B OVERFLOW? >>AC3F
AC3D NO, WOULD PROGRAM EXCEED HIMEM?
AC3F IF SO...
AC41 "PROGRAM TOO LARGE" >>AC14
AC43 ELSE, PICK UP LENGTH AGAIN (BEC8)
AC49 AND GO READ IT IN <AF98>
AC4C ERROR? >>AC14
AC4E CLOSE FILE <AF94>
AC51 ERROR? >>AC14
AC53 RELOCATE PROGRAM IF NECESSARY <AC61>
AC5C COPY AD PARM TO APPLESOFT PGM END PTR
AC60 RETURN

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

AC61 ---
AC62 WAS APPLESOFT PROGRAM SAVED FROM SAME
AC64 MEMORY LOCATION? (BE99)
AC73 YES, NOTHING TO DO THEN >>ACBA
AC79 ELSE, LOOP THROUGH PROGRAM
AC7B ADJUSTING ALL ADDRESSES TO
AC7D THE NEW LOAD LOCATION

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

AC97 WAS A LINE NUMBER PARM GIVEN? (BE57)
AC9D NO, NEVER MIND >>ACBA
AC9F COPY LINE KEYWORD VALUE TO APPLESOFT'S LINE # (BE68)
ACAF THEN CALL APPLESOFT TO FIND THE LINE <D61A>
ACB1 SUBTRACT ONE FROM THE ADDRESS
ACB3 AND POINT APPLESOFT'S GETCHR SUBROUTINE
ACB5 AT IT (SO NEXT CHAR READ WILL BE FIRST
ACB5 CHARACTER ON THE LINE).
ACBA RETURN

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

ACBE DOES FILE EXIST ALREADY? >>ACDF
ACBD NO, TYPE = BAS
ACBF IN T KEYWORD VALUE (BE6A)
ACCE AND MLI LIST (BE99)
ACCF ALLOW ALL ACCESSES (READ/WRITE/ETC.) (BE97)
ACCH SAVE PROGRAM START ADDRESS IN (BE55)
ACCF AUXID'S (BE99)
ACDA GO CREATE A NEW FILE <AD46>
ACDD ERROR? >>AD28

ACDF WRITE ACCESS REQUESTED
ACED BASE TYPE FILE
ACF OPEN IT <B194>
ACFH ERROR? >>AD28
ACF SUBTRACT APPLESOFT PTRS TO COMPUTE
ACGD LENGTH OF PROGRAM.
ACGE STORE THIS IN EOF MARK LIST (BEC8)
ACF8 MSB OF EOF MARK IS Ø (<64K PGM) (BECA)
AD00 POINT LIST TO PROGRAM AS DATA TO WRITE (BED7)
AD08 WRITE A RANGE TO DISK FILE <AF9C>
AD0B ERROR? >>AD28
AD0F MLI; SET EOF (TO TRUNCATE OLD LONGER FILE) <BE78>
AD12 ERROR? >>AD28
AD14 CLOSE THE FILE <AF94>
AD17 ERROR? >>AD28
AD1B DOES PROGRAM START MATCH AUXID IN FILE INFO?
AD20 NO, CHANGE IT >>AD29
AD28 ELSE, EXIT

AD29 TO CHANGE IT, (BE99)
AD2F EXIT THRU SET FILE INFO ROUTINE >>B7D9

ACD2 ********** "CREATE" COMMAND **********

ACD2 AUXID = 0 (AS OR RECLN)
ACD3 TYPE KEYWORD GIVEN?
ACD5 YES >>AD46
ACD3 NO, ASSUME TYPE = DIR (BE6A)

ACD6 *** CREATE FILE ENTRY *** (BE43)
ACD9 EXEC FILE ACTIVE?
ACDC HOW MANY FILES ARE OPEN INCLUDING EXEC? (BE4D)
ACDF 8 OR MORE?
ACD1 YES, ERROR >>AD60
ACD6 ELSE, SET TYPE IN MLI LIST (BEA4)
AD50 FULL ACCESS (READ/WRITE/ETC.)
AD5B KIND = STANDARD FILE
AD5D DIR FILE WANTED?
AD5F  NO >>AD63
AD61  YES, KIND = DIR FILE
AD63  SET ACCESS (BA24)
AD66  AND KIND (BE7A)
AD6B  MLI: CREATE (DON'T COMPARE) >>18 JUN 84 NEXT OBJECT ADDR: AD5F
AD6E  "RAM TOO LARGE" ERROR
AD70  RETURN

AD71  ******************** "RENAME" COMMAND
AD7F  --
AD75  SECOND PATHNAME GIVEN? >>BE70
AD76  IF SO, GO MLI: RENAME >>BACK HERE) >>BE70
AD7A  "SYNTAX ERROR" OTHERWISE
AD7D  ******************** "DELETE" COMMAND
AD7F  --
AD77  SETUP MLI: DELETE CALL TO ********************
AD7F  EXIT THRU MLI CALL >>BE70

AD82  ******************** "LOCK" COMMAND
AD87  --
AD82  GET FILE INFO FOR PATHNAME >>AE39
AD85  GET ACCESS CODES (BE79)
AD88  TURN OFF ALL... >> BE7D
AD89  BUT READ
AD8F  THEN GO SET UPDATED FILETYPE

AD92  ******************** "UNLOCK" COMMAND
AD97  --
AD92  GET FILE INFO FOR PATHNAME
AD95  TURN ON ALL FILE ACCESS >> <BE7D>
AD9D  THEN GO SET UPDATED FILETYPE

AD99  ******************** "PREFIX" COMMAND
AD9E  --
AD99  SLOT/DRIVE GIVEN ON COMMAND >>BE7F
AD9F  IF SO, GIT OPERAND ALREADY
ADAB  ELSE, (BE56)
ADAC  CHECK FOR PATHNAME = 4B1 <BE78>
ADAE  AND GO DO MLI: SET PREFIX >>BE7F
ADBF  ELSE, IS BASIC PROGRAM <<BE7F
ADB2  IF SO, SET PREFIX ACTIVELY
ADBF  NO, NEW LINE <9FA>
ADB3  END OF NAME YET? >>ADCC AND? (BE57)
ADB6  NO, COPY NAME IN PATHNAME >>ADAC
ADCC  TO OUTPUT DEVICE <9FA>
ADCD  AND SKIP A BLANK LINE
ADB3  DONE

AD1F  UNNAG? >>ADD1
AD1F  FLAG >>ADD1
AD1F  E1 BUFFER (BCBD)
AD1F  FAB

Beneath Apple ProDOS Supplement
BASIC Interpreter (BI) -- V1.1 -- 18 JUN 84
NEXT OBJECT ADDR: AE4D

ADDR DESCRIPTION/CONTENT

AE4D PICK OR EXIT -- V1.1 -- 18 JUN 84
NEXT OBJECT ADDR: AE4D

AE50 NET FROM L KEYWORD VALUE (BE5F)

AE52 B "NAUGHT"

AE59 Y "VEN?

AE5B E "NO, NAME OF (BE5D)"

AE5E TRUE "L = 7C"

AE63 COMP "FOR"

AE66 BLUE "SURE NO1" >> AE78

AE72 MArk "SE", "FAA" >> AE92

AE74 OR "VEN"

AE77 RET "INVR" >> AE72

AE78 ""MORROW OCCURED >> AE92

AE7B RET "ERROR"

AE7C --- GET EOF? >> AE97

AE7E ML "A99" >> AE99

AE80 ERROR "SOF NOT ERROR"

AE83 GET "NOT GROUP"

AE86 SET "A992"

AE88 NO "PROGRA" >> AE78

AE8E YES "UN"

AE90 --- "AK1" >> AE91

AE91 RET "ACED 64K (BEC8)"

AE92 STORE "WORD MS" >> AE92

AE99 B "ABC4" "TOO LARGE"

AE9D NO "COPY 3"

AEB1 YES "SET MAP"

AEB4 ML "ERROR? >> TO READ OR WRITE (BE99)

AEB6 NO "RANGES?

AEB8 E "LENGTH (BE99)"

AEBB BSA "A298"

AEE0 NO "FORCE ON (BE70)"

AEE2 ML "TRY SET A64"

AEC1 AND "IN ERROR?"

AEC3 RET "COMMAN" => AECD

AEC4 GET THE READ (LOAD/BRUNING)?

AEC7 AEE2 "??"

AEC9 BSA "READ IS CF FORWARD TO MARK <BE70"

AECB NO "T MARK AGAIN >> AEAA

AECF WRITE "READ OR AECD"

AEEF ML "R >> AE5"

AED2 ERROR >> AE4P "EXIT TH"

AED4 THE "CORRECT >> AECP

AED5 "WRITE <BE70"

AED7 "CLOSE >> AF94
BASIC Interpreter (BI) -- V1.1 -- 18 JUN 84

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>AF5D</td>
<td>FROM AUXID (BCBE)</td>
</tr>
<tr>
<td>AF63</td>
<td>ADJUST MSB OF THIS BY THE DIFFERENCE</td>
</tr>
<tr>
<td>AF66</td>
<td>BETWEEN HIMEM'S (NOW AND WHEN STORED) (BCBD)</td>
</tr>
<tr>
<td>AF73</td>
<td>MAKE SURE VARS WON'T OVERLAP PROGRAM</td>
</tr>
<tr>
<td>AF75</td>
<td>IF SO, ERROR &gt;&gt;AF90</td>
</tr>
<tr>
<td>AF7F</td>
<td>COMPUTE LENGTH OF ALL VARS/STRINGS</td>
</tr>
<tr>
<td>AF81</td>
<td>(HIMEM-START) (BCBF)</td>
</tr>
<tr>
<td>AF85</td>
<td>GO READ COMBINED VARS INTO MEMORY &lt;AF90&gt;</td>
</tr>
<tr>
<td>AF88</td>
<td>ERROR? &gt;&gt;AF39</td>
</tr>
<tr>
<td>AF8A</td>
<td>CLOSE THE FILE &lt;AF94&gt;</td>
</tr>
<tr>
<td>AF8D</td>
<td>EXIT BY REEXPANDING THE VARS DOWN &gt;&gt;AF32</td>
</tr>
<tr>
<td>AF90</td>
<td>&quot;PROGRAM TOO LARGE&quot; ERROR</td>
</tr>
<tr>
<td>AF93</td>
<td>RETURN</td>
</tr>
</tbody>
</table>

AF94 *********** CLOSE FILE ***************************************

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>AF94</td>
<td>SET MLI CLOSE OPCODE</td>
</tr>
<tr>
<td>AF96</td>
<td>AND GO TO MLI &gt;&gt;AfA4</td>
</tr>
</tbody>
</table>

AF98 ********** READ/WRITE A RANGE *******************************

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>AF98</td>
<td>READ MLI OPCODE</td>
</tr>
<tr>
<td>AF9A</td>
<td>JUMP IN &gt;&gt;AF9E</td>
</tr>
<tr>
<td>AF9C</td>
<td>WRITE MLI OPCODE</td>
</tr>
<tr>
<td>AF9E</td>
<td>STORE LENGTH (BEDA)</td>
</tr>
<tr>
<td>AF4D</td>
<td>EXIT THRU MLI:READ OR WRITE &gt;&gt;BE70</td>
</tr>
</tbody>
</table>

AF77 ********** "PR#" COMMAND ***************************************

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>AF77</td>
<td>USE CWSL AND OUTFEC</td>
</tr>
<tr>
<td>AFAC</td>
<td>JUMP TO COMMON CODE &gt;&gt;AFB5</td>
</tr>
</tbody>
</table>

AFAE ********** "IN#" COMMAND ***************************************

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFAE</td>
<td>USE KSWL</td>
</tr>
<tr>
<td>AFB3</td>
<td>AND INVEC</td>
</tr>
<tr>
<td>AFB5</td>
<td>OR IN SLOT GIVEN BY USER (B668)</td>
</tr>
<tr>
<td>AF89</td>
<td>*2 FOR USE AS INDEX INTO TABLE</td>
</tr>
<tr>
<td>AF8D</td>
<td>WAS SLOT PARAMETER GIVEN?</td>
</tr>
<tr>
<td>AF8F</td>
<td>NO... &gt;&gt;AFD2</td>
</tr>
<tr>
<td>AFD1</td>
<td>YES, (BE57)</td>
</tr>
<tr>
<td>AFD4</td>
<td>AD GIVEN? &gt;&gt;AFE7</td>
</tr>
<tr>
<td>AFD6</td>
<td>NO, GET OUTFEC OR INVEC FOR THIS SLOT (BE10)</td>
</tr>
<tr>
<td>AFD9</td>
<td>AND STORE ON AD KEYWORD VALUE (B556)</td>
</tr>
<tr>
<td>AF92</td>
<td>VALIDITY CHECK I/O DRIVER &lt;AF99&gt;</td>
</tr>
<tr>
<td>AF95</td>
<td>NO GOOD? &gt;&gt;AF6</td>
</tr>
<tr>
<td>AF67</td>
<td>GET INDEX TO CWSL OR KSWL (B6A9)</td>
</tr>
<tr>
<td>AF6D</td>
<td>AND REPLACE ONE OR THE OTHER WITH (B036)</td>
</tr>
<tr>
<td>AF89</td>
<td>HIS ADDRESS (BE59)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>AF66</td>
<td>RETURN</td>
</tr>
<tr>
<td>AF77</td>
<td>VALIDITY CHECK AD KEYWORD VALUE &lt;AF99&gt;</td>
</tr>
<tr>
<td>AF89</td>
<td>NO GOOD? &gt;&gt;AF8</td>
</tr>
<tr>
<td>AF9B</td>
<td>GOOD, COPY VALUE TO INVEC OR OUTFEC (B659)</td>
</tr>
<tr>
<td>AF9F</td>
<td>EXIT BUT DON'T REDIRECT I/O NOW</td>
</tr>
</tbody>
</table>

AF99 *********** VALIDITY CHECK I/O DRIVER **************************

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>AF99</td>
<td>$3A/3B --&gt; NEW HANDLER (FROM AD PARM) (BE58)</td>
</tr>
<tr>
<td>B005</td>
<td>IS DRIVER IN MAIN RAM (BE30)</td>
</tr>
<tr>
<td>B007</td>
<td>YES &gt;&gt;B01E</td>
</tr>
<tr>
<td>B009</td>
<td>NO, RESET I/O CARD ROMS (CFFP)</td>
</tr>
<tr>
<td>B00C</td>
<td>USE $3C TO COUNT ITERATIONS</td>
</tr>
<tr>
<td>B00E</td>
<td>TEST ROM AT USER'S ADDRESS</td>
</tr>
<tr>
<td>B014</td>
<td>FOR STABILITY</td>
</tr>
<tr>
<td>B018</td>
<td>256 TIMES</td>
</tr>
<tr>
<td>B01C</td>
<td>MUST BE OK</td>
</tr>
<tr>
<td>B01D</td>
<td>RETURN</td>
</tr>
<tr>
<td>B01E</td>
<td>MAIN RAM I/O DRIVER</td>
</tr>
<tr>
<td>B020</td>
<td>MUST START WITH A &quot;CLD&quot; INSTRUCTION</td>
</tr>
<tr>
<td>B022</td>
<td>OK... &gt;&gt;B01C</td>
</tr>
<tr>
<td>B024</td>
<td>ELSE, &quot;NO DEVICE CONNECTED&quot;</td>
</tr>
<tr>
<td>B027</td>
<td>RETURN</td>
</tr>
</tbody>
</table>

B028 *********** "BYE" COMMAND ***************************************

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>B028</td>
<td>CLOSE ANY OPEN FILES &lt;B4F2&gt;</td>
</tr>
<tr>
<td>B028</td>
<td>CLOSE EXEC FILE, IF ANY &lt;B2FB&gt;</td>
</tr>
<tr>
<td>B038</td>
<td>MLI CALL: &lt;BF00&gt;</td>
</tr>
<tr>
<td>B033</td>
<td>QUIT</td>
</tr>
<tr>
<td>B034</td>
<td>USE READ PARMLIST BECAUSE QUIT DOESN'T NEED PARMS</td>
</tr>
</tbody>
</table>

B036 *********** "CAT" COMMAND ***************************************

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>B036</td>
<td>39 CHARACTERS PER LINE</td>
</tr>
<tr>
<td>B038</td>
<td>THEN PROCESS LIKE &quot;CATALOG&quot; &gt;&gt;B03C</td>
</tr>
</tbody>
</table>

B03A *********** "CATALOG" COMMAND ***************************************

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>B03A</td>
<td>79 CHARACTERS PER LINE</td>
</tr>
<tr>
<td>B03C</td>
<td>STORE LINE LENGTH (BC66)</td>
</tr>
<tr>
<td>B042</td>
<td>TEST FOR T AND</td>
</tr>
<tr>
<td>B044</td>
<td>...PATHNAME1 GIVEN</td>
</tr>
<tr>
<td>B045</td>
<td>GOT T &gt;&gt;B04A</td>
</tr>
<tr>
<td>B047</td>
<td>K T, P=O (ANY TYPE WILL DO) (B65A)</td>
</tr>
<tr>
<td>B04A</td>
<td>GOT PATHNAME1 &gt;&gt;B051</td>
</tr>
<tr>
<td>B04C</td>
<td>NO PATHNAME1, GET FILE INFO FOR PREFIX &lt;BD0&gt;</td>
</tr>
<tr>
<td>B04F</td>
<td>ERROR? &gt;&gt;B0B7</td>
</tr>
<tr>
<td>B051</td>
<td>OPEN/READ DIRECTORY HEADER &lt;B14A&gt;</td>
</tr>
</tbody>
</table>
B054  ERROR? >>B0B7
B056  SKIR TO A NEW LINE <9FAB>
B059  FORMA DIRECTORY'S NAME TO
B05C  PRINT $201 <9F9D>
B05F  SKIR TO A NEW LINE <9FAB>
B062  BLANK $201 BUFFER <A66C>
B067  UNPACK HEADER MESSAGE LINE $201 <9F9D>
B06A  PRINT IT (48 OR 80 COLUMNS)
B06D  SKIR AS A NEW LINE <9FAB>
B073  ANY FILES IN THIS DIRECTORY
B076  NO >>BBA3
B078  YES, READ NEXT ENTRY <<BD11
B07B  ERROR? >>B0B7
B07D  GET TYPE REQUESTED FOR SEARCH? <<BCB8
B088  ANY TYPE WILL DO? >>B0B7
B082  NO, CHECK TYPE AGAINST THIS
B085  NOT IF, SKIR IT >>B0B0
B087  ELSE FORMAT ENTRY TO $201 OR $200
B08A  AND PRINT $201 <9F9D>
B08D  CHECK KEYBOARD (C060)
B090  FOR A CONTROL-C
B092  IGNORE ANYTHING ELSE >>B0BE
B094  CONTROL-C, WHAT STATE ARE WE
B097  DEFERR >>B0A3
B099  NO, IMMEDIATE, RESET KEYBOA
B09C  AND EXIT RIGHT NOW >>B0A3
B09E  ELSE, ANY FILES LEFT IN COU
B0A1  YES, CONTINUE >>B078
B0A3  ELSE, CLOSE DIRECTORY <<AF94
B0A6  ERROR? >>B0B7
B0A8  SKIR TO A NEW LINE <9FAB>
B0A9  FORMAT BLOCKS FREE AND IN USE
B0B6  ERROR? >>B0B7
B0B8  PRINT $201 <9F9D>
B0B9  SKIR A LINE <9FAB>
B0BC  DSH

B0B8  FORMAT NAME OF DIR

B0B8  BLACK $201 BUFFER <A66C>
B0BD  FILE NAME IS AT +1 INTO DIR
B0BE  GET NAME LENGTH/TYPE (825D)
B0B2  VOLUME DIRECTORY HEADER?
B0C4  NO >>B0CA
B0C6  YES, START NAME WITH "/*" (0
B0CA  --
B0CB  ISOLATE NAME LENGTH FROM TY
B0CD  AND GET UP LENGTH TO COPY (~
B0D2  COPY DIRECTORY NAME TO (825B

B0DB  BLACK $201 BUFFER <A66C>
B0DF  FILE NAME IS AT +1 INTO DIR
B0E0  GET NAME LENGTH/TYPE (825D)
B0E2  VOLUME DIRECTORY HEADER?
B0E4  NO >>B0CA
B0E6  YES, START NAME WITH /* (0
B0EC  --
B0EB  ISOLATE NAME LENGTH FROM TY
B0ED  AND GET UP LENGTH TO COPY (~
B0EE  COPY DIRECTORY NAME TO (825B

NEXT OBJECT ADDR: B0D7
BASIC Interpreter (B1) -- V1.1 -- 10 JUN 84  NEXT OBJECT ADDR: B193

B194 ********** OPEN FILE **************************************
A REGISTER = ACCESS BITS
X REGISTER = DEFAULT TYPE

B194 ---
B195 T KEYWORD GIVEN?
B19A NO >> B19F
B19C YES, USE KEYWORD VALUE INSTEAD (BE6A)
B19F ---
B1A0 EXISTING FILE OF THIS TYPE? (BE88)
B1A3 NO, ERROR >> B1C9
B1A5 CHECK ACCESS REQUESTED (BE77)
B1A6 REQUESTED ACCESS NOT PERMITTED >> B1CD
B1A8 SET SYSTEM BUFFER IN OPEN PARM LIST (BC88)
B1B2 LEVEL = $3F (BF94)
B1B7 MLI: OPEN <BE70>
B1B9 ERROR? >> B1CB
B1BF SAVE REFPNUM IN READ/ WRITE PARMLIST (BE66)
B1C2 AND CLOSE PARMLIST (BEDE)
B1C5 AND GET/SET EOF/MARK LIST (BE77)
B1C8 AND EXIT

B1C9 "FILE TYPE MISMATCH"
B1CC RETURN
B1CD "FILE LOCKED"
B1DD RETURN

B1D1 ********** READ NEXT DIRECTORY ENTRY *************

B1D1 FORCE MARK TO START OF THIS BLOCK (BE99)
B1D9 CHECK ENTRY NUMBER (BC88)
B1DE LAST ENTRY IN THIS BLOCK? (BC88)
B1E1 NO >> B1ED
B1E4 YES, ENTRY 0 NEXT TIME (BCBB)
B1E7 BUMP MARK TO NEXT BLOCK (BEC9)
B1ED ---
B1EF MARK POSITIONED TO PROPER ENTRY YET? >>B1F6
B1F1 NO, BUMP POINTER TO NEXT ENTRY (BC77)
B1F4 AND CONTINUE IP STILL FIRST PAGE >>B1ED
B1F6 JUST ENTERED SECOND PAGE >>B1BA
B1FB ADD 4 TO PTR TO ADJUST FOR BLOCK PREFIX
B1FF MLI: SET MARK <BE70>
B202 ERROR? >> B21D
B206 MLI: READ <BE70>
B209 ERROR? >> B21D
B20B BUMP ENTRY COUNTER (BC88)
B211 IS THIS ENTRY VALID?
B213 NO, SKIP OVER IT >> B1D1
B215 DECREMENT FILE COUNT (BEC9)
B21D AND RETURN TO CALLER
---

**BASIC Interpreter (B1) -- V1.1 -- 18 JUN 84 NEXT OBJECT ADDR: B278**

---

**ADDR DESCRIPTION/CONTENTS**

---

--- EXEC TOPMOST BUFFER *******

****** MAKE ******

B27A OTHERS ARE OPEN PAGES PER BUFFER

B27C OPENCOUNT*4 (4 BUFFER TO FIND TOP BUFFER (BC9B)

B27E ADD THIS TO MY BS TO FIND THE FILE WHICH (BC93)

B282 SEARCH OPEN FILE... >>B28B

B285 IS USING THIS IN B285, BREAK!

B28A IF IT IS NOT FO...

B28B --- TO THE NEW BUFFER INSTEAD (BC93)

B28C MOVE THAT FILE >>B28F AND ALSO (BC9B)

B28F GET THAT FILE >>B278

B297 MLI: SET BUFF <<0

B29A NO ERRORS? >>B29B

B29B IF ERROR, BREAK

B29D --- NEW EXEC FILE *******

****** OPEN ******

B29E SET NEW BUFFER T FOR EXEC TOO (BECF)

B2A1 SET UP OPEN LIST

B2A6 LEVEL = 0 (BF94:FL2) <BE70>

B2A8 MLI: OPEN (EXEC)

B2AB NO ERROR? >>B2AE

B2AE BUFFER FIRST >>A24C

B2B1 IF ERROR, FREE ERROR

B2B6 THEN EXIT WITH EXEC (BECF)

B2BB SAVE BUFSIZE FOR #ED6

B2BD AND REFNUM TOO

B2CE EXEC COMMAND *******

****** COMP ******

B2C3 SAVE READ REFNUM <<0 (BEC7)

B2C6 AND GET/SET REFNUM >>0 (BED2)

B2CE AND NWLINE REFNUM AUXID (BE5F)

B2CF SET "L" VALUE EQU TO OPEN FILE TABLE <<38B

B2D8 SAVE PATHNAME/FIND LINE CHARS (BED1)

B2DD IGNORE MSN FOR FILE <B270>

B2D2 MLI: SET NEWLINES ON COMMAND LINE?

B2DA WAS "P" OR "O"

B2DE NO >>B2DA

B2E1 CHANGED STARTING PT <B22A

B2EC YES, POSITION TP4

B2EF NO ERRORS? >>B2EF EXEC >>B245

B2F1 IF ERROR, GO CFE

B2F4 MARK EXEC ACTIVALLER

B2FA AND RETURN TO C

---

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

B2FB EXEC ACTIVE? (BE43)

B2FE NO, SKIP IT >>B30B

B305 Indicate EXEC FILE CLOSING (BE4B)

B30B Pick up REFNUM FOR EXEC (BC9B)

B30F RETURN

---

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

B30C FILE NOT FOUND? >>B347

B31F FILE FOUND, WAS A PATHNAME1 GIVEN?

B313 YES >>B31D

B315 NO,

B319 PRINT "FILE "APPLE COMPUTER..." <9F8C>

B31A AND A NEW LINE <9FAB>

B31B THEN EXIT

B31E RETURN

---

--- B31F ********** FLUSH ALL OPEN FILES *******

B31F REFNUM = 0 (ALL FILES)

B321 JUMP INTO FLUSH >>B32F

---

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

B323 ---

B326 WAS PATHNAME1 GIVEN?

B32B NO, FLUSH ALL FILES >>B32F

B32A ELSE, LOOK UP NAME IN OPEN FILE LISTS <<41F

B32D NOT AN OPEN FILE >>B317

B32F SAVE REFNUM IN PARM LIST (BEDE)

B334 MLI: FLUSH <BE70>

B337 EXIT

---

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

B338 ---

B339 LOOK UP NAME IN OPEN FILE LIST <B41F

B33C NOT CURRENTLY OPEN? >>B34B

B33E ---

B33F IT IS OPEN, "FILE BUSY" ERROR

B342 RETURN
BASIC Interpreter (B1) -- V1.1 -- 18 JUN 84  NEXT OBJECT ADDR: B342

ADDR  DESCRIPTION/CONTENTS

B343  "FILE TYPE MISMATCH" ERROR
B346  RETURN
B347  "PATH NOT FOUND" ERROR
B349  ---
B34A  RETURN
B34B  ---
B34C  ASSUME "L" IS ZERO
B34D  WAS "L" KEYWORD GIVEN?
B34E  YES, USE HIS VALUE >> B35D
B357  NO, SET "L" TO ZERO (BE60)
B360  WAS "T" GIVEN?
B364  YES, USE HIS TYPE >> B36B
B366  ELSE, DEFAULT TO "TAX"
B368  DOES THE FILE ALREADY EXIST? >> B3BE
B36D  NO, "T" GIVEN? IF SO, ERROR >> B347
B36F  FORCE TYPE = "TXT" (BE60)
B37A  COPY "L" KEYWORD VALUE (BE5F)
B37B  TO CREATE (BE66)
B380  AND SET FILE INFO LISTS (BE6A)
B389  GO CREATE THE FILE << AD40
B38C  ERROR? >> B349
B38E  CHECK FILE TYPE (BE68)
B391  AGAINST HIS "T" VALUE (BE6A)
B394  MISMATCH? >> B343
B396  NO, TYPE = TXT?
B39B  NO >> B3AD
B39A  YES, GET RECORD LENGTH FROM AUXID (BE6B)
B3A3  WAS "L" KEYWORD VALUE GIVEN?
B3A5  YES, USE THAT INSTEAD >> B3AD
B3A7  OTHERWISE, SAVE AUXID RECORD LEN (BE6D)
B3AD  ALLOCATE A NEW FILE BUFFER << A1F5
B3B0  ERROR? >> B349
B3B2  GET BUFFER PAGE NO. (BC88)
B3B8  AND STORE IN OPEN LIST (BCEF)
B3BA  LEVEL = 7 (BF94)
B3BF  MLIRE OPEN << BE70
B3C2  NO ERRORS? >> B3CB
B3C4  ---
B3C5  ERROR, FREE BUFFER FIRST << A24C
B3CA  THEN EXIT WITH ERROR CODE
B3CB  CHECK FILE TYPE AGAIN (BE6B)
B3CE  "DIR" FILE?
B3D0  YES >> B3D3
B3D2  NO
B3D3  ---
BASIC Interpreter (B1) -- V1.1 -- 18 JUN 84
**NEXT OBJECT ADDR: B446**

---

**B446**

CAN'T FILE

**B448** NO, THEN

**B450** IS HE LOOKING FOR AN EXEC FILE? 

**B452** YES, EXEC <<B454 >>

**B45C** AND RETURN FILE CLOSING (BE4E)

**B45E** "FILE NOT" ERROR

**B461** ERROR COBOL

---

**B462**

COMPARE FILENAMES

**B468** REPNUM ERROR FILENAME INDEX

**B470** SAME LENGTH FLAG FROM THIS ENTRY (BCFE)

**B473** NO, CAN'T BE HIS FILENAME? (0280)

**B474** MAKE SURE IT THEN >>B498

**B47A** IF DEBUG LENGTH DOES NOT EXCEED 29

**B47C** USE 3A ERROR ONLY LOOK AT FIRST 29

**B481** COPY "FILE LOOP COUNTER

**B48A** ---

**B48B** COMPLETE THIS FILE TO KEYWORD (BCA4)

**B491** NO MATCH ERROR (0280)

**B498** MATCH, EXIT WITH FLAG CLEAR >>B498

**B499**

CLOSE "COMMAND

**B49C** PATHNAME

**B49E** NO, CLOSE GIVEN?

**B4AE** YES, LOOK ALL FILES >>B4F2

**B4AF** NOT FOUND >>B41F

**B4A1** FOUND IT >>B441

**B4A7** MARK BUFFER STORE REPNUM IN CLOSE LIST (BEDE)

**B4AE** EXEC CLOSE PAGE FREE (BC88)

**B4BE** YES ...NO >>B4E7

**B4BE** GET OPENED NEED TO COMPRESS LISTS >>B4CF

**B4BF** SWAP BUFFER (LAST OPENED FILE NO.) (BE4D)

**B4C5** AND REPNUMS (BC93)

**B4CF** YES WITH THE LAST OPENED FILE (BC9B)

**B4D0** LEVEL = 1

**B4D6** MLI: CLOSE (BEF94)

**B4D9** ERROR? >>B760

**B4DB** RELEASE >>B429

**B4DE** EXEC FILE BUFFER <<A24C>

**B4E1** NO >>B4E2 "ALSO" ERROR?

**B4E5** YES, EXIT

**B4E9** AND NO LONGER ACTIVE (BE43)

**B4ED** RETURN INSIDE CLOSING (BE4E)

---

BASIC Interpreter (B1) -- V1.1 -- 18 JUN 84
**NEXT OBJECT ADDR: B446**

---

**B44E** 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 ALL OVER AGAIN

**B502** EXIT WHEN ALL ARE CLOSED

---

**B503**

**B505** SET CLOSE REPNUM TO ZERO (ALL FILES)

**B50A** LEVEL = 7 (LEVEL 0 FILES ALREADY CLOSED)

**B50E** EXIT THRU MLI: CLOSE >>B7E0

---

**B512**

"POSITION" COMMAND

**B512**

LOOKUP NAME OF FILE <<B41F>

**B515** NOT OPEN? >>B57F

**B517** SET REPNUM IN READ/WRITE PARMLIST (BED1)

**B519** AND SET NEWLINE LIST (BED2)

**B51D** DIR FILE2 (BE47)

**B520** YES, GET OUT RIGHT NOW! >>B590

---

**B522** "R" OR "E" GIVEN? (BE57)

**B527** NO, INVALID VARY >>B75D

**B529** BOTH GIVEN

**B52B** YES, INVALID PARM >>B57D

**B52D** JUST "R" GIVEN

**B52F** NO, JUST "E" >>B53D

**B531** JUST "R", COPY "R" VALUE TO "R" (BE57)

**B534** "R" AND "E" ARE ALIASES (BE5D)

**B53D** SET COUNT TO 239. (MAXIMUM LINE LEN

**B54C** BUFFER IS AT $200 (BED9)

**B54F**

**B551** NEW LINE CHAR IS EITHER $0D OR $0D

**B556** MLI: SET NEWLINE <<BE70>

**B559** ERROR? >>B57F

---

**B552**

**B555** "R" = W? (BE56)

**B562** YES, DONE >>B590

**B564** ELSE...

**B566** MLI: READ NEXT FIELD (LINE) <<BE70>

**B569** ERROR? >>B57F

**B56D** DECREMENT "F" VALUE BY ONE

---

**NEXT OBJECT ADDR: B44E**
BASIC Interpreter (B1) -- V1.1 -- 18 JUN 84 NEXT OBJECT ADDR: B57B

B57B AND GO CHECK IT AGAIN >>B55B

B57D "INVALID PARAMETER" ERROR

B57F ---

B580 EXIT TO CALLER

B581 *********** COMPUTE NEW FILE POSITION ***********************

(Computes absolute file position mark)

B581 ACCUM = CURRENT RECORD LENGTH (BCA4)
B595 MARK = 0 (BEC8)

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

B59E SHIFT "R" VALUE RIGHT (BB66)
B5A5 IF LOW BIT OFF, NO ADD >>B5BF
B5A9 ADD ONE INSTANCE OF RECLEN TO MARK (BCAF)
B5B6 OVERFLOW? >>B5D2
B5BD ACCUM OVERFLOW? >>B5D2
B5BF SCALE ACCUM (MULTIPLIER) UP BY 2 (BCAF)
B5CB IF "R" NOT ZERO... (BB03)
B5CE CONTINUE LOOPING >>B59E
B5D1 ELSE, EXIT TO CALLER

B5D2 "RANGE ERROR"

B5D5 RETURN

B5D6 *********** "READ" COMMAND ***********************

B5D6 LOOK UP FILE NAME <B41F>
B5D9 NOT OPEN? >>B62B
B5DB ITS OPEN, STORE REFNUM IN READ/WRITE... (BED6)
B5DE GET/SET... (BE07)
B5E1 AND SET NEWLINE PARMLISTS (BED2)
B5E4 DIR FILE? (BE47)
B5E7 YES, SPECIAL HANDLING REQUIRED >>B62C
B5E9 NO, PRE-POSITION FOR "R", "F", OR "R" <B666>
B5EC ERROR POSITIONING? >>B62B
B5EE ASSUME "L" = 239.
B5F5 "L" GIVEN?
B5F7 NO >>B64C
B5F9 YES, USE HIS "L" VALUE (B5EF)
B5FF UNLESS ITS >256 >>B661
B603 OR >239. >>B601
B607 DOUBLE QUOTED IT SO COMMAS COME THRU (B20U)
B60A READ INTO 230L
B60C IF NO "L", READ TO $200 (BED7)
B612 NL CHAR = $00/$8D (OR NONE IF "L") (BED3)
B621 MLI: SET NEWLINE <BE70>
B624 ERROR? >>B62B
B626 ---

BASIC Interpreter (B1) -- V1.1 -- 18 JUN 84 NEXT OBJECT ADDR: B62B

B628 MARK INPUT "READ" FILE ACTIVE (BE44)
B62B AND RETURN

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

B62C SET READ/WRITE LIST REFNUM (BED6)
B62F AND GET/SET LIST REFNUM (BE07)
B634 READING TO $209 (BED7)
B63E INIT CAT FLAG TO FIRST LINE VALUE (BE4F)
B644 "R" GIVEN?
B647 NO, DONE >>B626
B64B YES, ZERO OUT MARK (BEC9)
B656 MLI: 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? >>B6B0
B67B "F" GIVEN? (BE57)
B680 NO >>B6B7
B682 SKIP LINES UNTIL "F" = 0 <B53D>
B685 ERROR? >>B6B0
B687 "B" GIVEN? (B57)
B68C NO >>B6A8
B690 MLI: GET MARK <BE70>
B693 ERROR? >>B6B0
B699 ADD "B" VALUE TO CURRENT MARK (BE5A)
B69C (3 BYTE ADD) (BEC8)
B6A6 OVERFLOW? >>B661
B6A8 MLI: READ FILE <BE70>
B6AD ERROR? >>B680
B6AF ---
B6B0 ---
B6B2 EXIT TO CALLER
B6B3  ********** "WRITE" COMMAND **********************

B6B3  LOOKUP OPEN FILE NAME <B41F>
B6B6  NOT AN OPEN FILE? >>B6C8
B6BB  STORE READ/WRITE REFNUM (BED6)
B6BB  AND GET/SET REFNUM (BEC7)
B6BE  AND NEWLINE REFNUM IN PARM LISTS (BED2)
B6C1  DIR FILE? (BE47)
B6C4  NO, OK >>B6CA

B6CA  YES, "FILE LOCKED" ERROR
B6CB  ---
B6C9  EXIT TO CALLER

B6CA  DATA BUFFER AT $200
B6B4  PRE-POSITION FOR "R", "F", AND "R" <B666>
B6B7  NO ERRORS? >>B6ED
B6B9  WAS ERROR A RANGE ERROR?
B6BB  NO, REAL ERROR >>B6C8
B6DD  YES, MY RANGE ERROR OR MLI'S?
B6DF  MINE... >>B6C8
B6E1  MLI'S...SET EOF FARTHER INTO FILE
B6E3  MLI: SET EOF <BE70>
B6E5  ERROR? >>B6C8
B6EB  AND THEN TRY AGAIN TO SET MARK <B676>
B6EB  ERROR? THEN I GIVE UP >>B6C8
B6ED  BUFFER IS AT HIMEM
B6EF  INDICATE OUTPUT "WRITE" FILE ACTIVE (BE45)
B6FD  RETURN TO CALLER

B6FE  ********** "APPEND" COMMAND **********************

B6FE  ---
B6FF  LOOK UP NAME IN OPEN FILE LIST <B41F>
B702  FOUND IT? >>B710
B705  NO, OPEN IT FIRST <B38B>
B708  ERROR? >>B71E
B70A  NO, REFNUM NON-ZERO? (BED0)
B70D  YES, OK >>B711
B70F  ELSE, BREAK111
B710  ---
B711  REFNUM TO READ/WRITE PARM LIST (BED6)
B714  AND GET/SET LIST (BEC7)
B717  DIR FILE? (BE47)
B71A  NO >>B720
BASIC Interpreter (BI) — V1.1 — 18 JUN 84  NEXT OBJECT ADDR: B7D7

ADDR  DESCRIPTION/CONTENTS

B7D9  ********** SET FILE INFO ****************************

B7D9  MODIFIED TIME/DATE = 0
B7E7  SET NUMBER OF PARMS (7)
B7EC  MLI CODE FOR SET FILE INFO
B7EE  EXIT THRU MLI: GET/SET FILE INFO >>BE78

B7F1  ********** BI I/O INDICATION VECTORS ***************

B7F1  DOGOUT VECTOR >>BE38
B7F4  DOBIN VECTOR >>BE3A

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

B7F7  IMMEDIATE MODE (STATE=0) CSW/CSWL
B7FB  DEFERRED MODE (STATE=4) CSW/CSWL
B7FF  (STATE=8) CSW/CSWL
B803  (STATE=C) CSWL

B805  ********** SYSCTBL ****************************

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

B805  CREATE: $A0  DESTROY: $A0  RENAME: $A0
B808  SFI:    $B4  GFI:    $B4  ONLINE:     $C6
B80B  SPPX:  $AC  OPPX:  $AC  OPEN:      $CB
B80E  NEWLINE:$D1  READ:  $D5  WRITE:     $D5
B811  CLOSE:  $DD  FLUSH:  $DD  SMARK:    $C6
B814  GMARK: $C6  SCEOF: $C6  GEOF:      $C6
B817  SCBU:  $C6  GBUF:  $C6

B819  ********** APPLESOFT TOKENS ********************

TOKENS REQUIRING SPECIAL ATTENTION HAVE
THEIR MSB OFF AND ARE AN OFFSET FROM A
JMP IN THE TRACE HANDLER IN THE BI

B819  FIRST IS $80 (END)
B823  CALL
B833  TRACE, NOTRACE, NORMAL
B837  INVERSE, FLASH
B83F  RESUME
B844  LET, IF
B853  PRINT, LIST

B859  ********** COMMAND NAME TABLES ***************

OFFSETS TO LAST CHARACTER OF EACH COMMAND
NAME IN THE COMMAND NAME TABLE BELOW.
COMMANDS ARE ARRANGED ACCORDING TO LENGTH
WITH THREE BYTE NAMES FIRST. IF THE MSB
OF AN INDEX IS ON, THEN THIS IS THE LAST

NAME OF THE GIVEN LENGTH (NEXT WILL BE
ONE BYTE LONGER).

B859  0  IN#    0  PR#    0  CAT
B85C  0  C  0  BYE    0  RUN
B85F  0  BRM  0  EXEC   0  LOAD
B862  0  LOCK  0  OPEN   0  READ
B865  0  SAVE  0  LOAD    0  BSAVE
B868  0  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  'VERIFYFILEDELETEFILECATALOGOPEN'
B879  'WRITEEXCREATENOTRESTORERENAMEBRUNLO'
B888  'EXECCHAIN#FLUSHREAPositionMONOP98'
B888  'PREFIXCLOSEAPPEND'

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

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

B889  (EXTERNAL)
B893  TAB
B895  ERK
B899  CRF
B89F  ERS
B8A3  EOF
B8A7  BRM
B8AA  ELC
B8AF  LOAD
B8B0  LOCK
B8B4  OPEN
B8B8  READ
B8BD  SAVE
B8BE  LOAD
B8C7  BSAVE
B8CD  CHAIN
B8C9  CLOSE
B8CF  FLUSH
B8D0  NOMON
B8D1  FORCE
B8D3  WRITE
B8D5  APPEND
B8D7  CREATE
B8D9  PREFIX
B8DB  RENAME
BASIC Interpreter (BI) -- V1.1 --18 JUN 84
NEXT OBJECT ADDR: B91F

B91F UNLOCK
B921 VERIFY
B923 CATALOG
B925 *RESTORE
B927 POSITION
B929 "-" COMMAND

B92B ************** PERMITTED KEYWORDS FOR CMDS ***************

TWO BYTES PER COMMAND IN THE ORDER ABOVE.
EACH ENTRY HAS 16 BIT SETTINGS FOR THE
PARAMETERS PERMITTED ON THAT COMMAND.

B940 = FETCH PREFIX, PATHNAME OPTIONAL
B940 = SLOT (FOR PR# OR IN#)
B960 = DEFERRED COMMAND ONLY
B970 = FILENAME IS OPTIONAL
B980 = IF FILE NOT FOUND, CREATE IT
B990 = "A" (ADDRESS) PERMITTED
B9A0 = "B" (BYTE) PERMITTED
B9B0 = "E" (END ADDRESS) PERMITTED
B9C0 = "L" (LENGTH) PERMITTED
B9D0 = "N" (LINE NO.) PERMITTED
B9E0 = "S" AND/OR "D" (SLOT/DRIVE)
B9F0 = "F" (FIELD) PERMITTED
B9C0 = "R" (RECORD) PERMITTED
("V" IS IGNORED)

B930 IN# . . X . . X . X
B931 KM . . . . . . . .
B932 CR . . . . . . . .
B933 EX . . . . . . . .
B934 RN . . . . . . . .
B935 BU . . . . . . . .
B936 RUN . . . . . . . .
B937 BR . . . . . . . .
B938 EXE . . . . . . . .
B939 LOAD . . . . . . . .
B93A LOCK . . . . . . . .
B93B OPEN . . . . . . . .
B93C READ . . . . . . . .
B93D SAVE . . . . . . . .
B93E BLOAD . . . . . . . .
B93F BSAVE . . . . . . . .

B949 CHAIN . . . . . . . X . . . . .
B94B CLOSE . . . . . . . X . . . . .
B94D FLUSH . . . . . . . X . . . . .
B94F NOMON . . . . . . . X . . . . .
B950 STORE . . . . . . . X . . . . .
B951 WRITE . . . . . . . X . . . . .
B952 APPEND . . . . . . . X . . . . .
B953 CREATE . . . . . . . X . . . . .
B954 DELETE . . . . . . . X . . . . .
B955 PREFIX . . . . . . . X . . . . .
B956 DELETE . . . . . . . X . . . . .
B957 RESTORE . . . . . . . X . . . . .
B958 POSITION . . . . . . . X . . . . .
B959 "-" . . . . . . . X . . . . .

B960 *************** KEYWORD NAME TABLE ***********************
B961 'ABELSROV'

B965 *************** KEYWORD BIT POSITION TABLE ******************
BIT POSITIONS IN PERMITTED PARMS TABLE
FOR EACH KEYWORD IN THE ORDER GIVEN IN
NAME TABLE. "V" IS 00 (NOT USED)

B975 ----

B97F *************** KEYWORD SIZE/OFFSET TABLE ******************
LOW 2 BYTES - SIZE=1 OF VALUE IN BYTES
HIGH 6 BITS- OFFSET TO LAST BYTE OF VALUE
FROM $BE58

B97F A: 2 BYTES AT +1
B970 B: 3 BYTES AT +4
B971 E: 2 BYTES AT +6
B972 L: 2 BYTES AT +8
B973 S: 1 BYTE AT +9
B974 D: 1 BYTE AT +A
B975 F: 2 BYTES AT +C
B976 R: 2 BYTES AT +E
B977 V: 1 BYTE AT +10 (IGNORED)
B978 Q: 2 BYTES AT +11

B979 *************** FILE TYPES TABLES ***********************
FILE TYPE CODES, GIVEN IN INVERSE ORDER
TO FILE TYPE NAMES WHICH FOLLOW.
BASIC Interpreter (BI) --  V1.1  --18 JUN 84  NEXT OBJECT ADDR: B989

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>B989</td>
<td>$FF = &quot;SYS&quot;</td>
</tr>
<tr>
<td>B98A</td>
<td>$FE = &quot;REL&quot;</td>
</tr>
<tr>
<td>B98B</td>
<td>$FD = &quot;VAR&quot;</td>
</tr>
<tr>
<td>B98C</td>
<td>$FC = &quot;BAS&quot;</td>
</tr>
<tr>
<td>B98D</td>
<td>$FB = &quot;INT&quot;</td>
</tr>
<tr>
<td>B98E</td>
<td>$FA = &quot;CMD&quot;</td>
</tr>
<tr>
<td>B98F</td>
<td>$F0 = &quot;DIR&quot;</td>
</tr>
<tr>
<td>B990</td>
<td>$0F = &quot;BIN&quot;</td>
</tr>
<tr>
<td>B991</td>
<td>$0E = &quot;TXT&quot;</td>
</tr>
<tr>
<td>B992</td>
<td>$0D = &quot;ASP&quot;</td>
</tr>
<tr>
<td>B993</td>
<td>$0C = &quot;AWP&quot;</td>
</tr>
<tr>
<td>B994</td>
<td>$1A = &quot;ASH&quot;</td>
</tr>
<tr>
<td>B995</td>
<td>$1B = &quot;ADB&quot;</td>
</tr>
<tr>
<td>B996</td>
<td>$19 = &quot;ADB&quot;</td>
</tr>
</tbody>
</table>

B997 'ADBASPAWPPASTXTBINFDIRCMDINTIVRBASVARRELSYS'

B9C1 ********** MONTH TABLE *******************************

B9C1 'JANFEBMARAPRMAYJUNJULAGSEPONOVDEC'

B9E5 '<NO DATE>'

B9EE ********** MLERRBL *******************************

ML ERROR CODES WHICH HAVE BI EQUIVALENTS

B9EE ---

BA01 ********** BIERRBL *******************************

BI EQUIVALENTS TO MLI ERROR CODES ABOVE
(IF MLI CODE NOT FOUND, MAPS TO LAST CODE
IN THIS TABLE, $08 "I/O ERROR")

BA01 ---

BA15 ********** INDEXES TO PACKED MESSAGES ******************

BY BI ERROR NUMBER

BA15 ---

BA29 ********** COMMON LETTERS IN MESSAGES ******************

BA29 'ACDEFLMNORTU'

BA38 ********** LESS COMMON LETTERS **************************

BA38 ---

BA39 'BGHKPSWKY/().:

BA48 ********** PACKED MESSAGES ******************************

BA48 "COPYRIGHT APPLE COMPUTER"

BA58 " NAME 

BA5B TAB($10)

BA5D "TYPE BLOCKS 

BA66 TAB($1E)

BA68 "MODIFIED"

BA6C TAB($2F)

BA7E "CREATE0 

BA74 "ENDFILE SUBTYPE"

BA7E "BLOCKS FREE:"

BA66 TAB($16)

BA88 "BLOCKS USED:"

BA91 TAB($2C)

BA93 "TOTAL BLOCKS:

BA9C "RANGE ERROR"  ERROR=$2

BA99 "NO DEVICE CONNECTED" ERROR=$3

BA9E "WRITE PROTECTED" ERROR=$4

BA97 "END OF DATA"  ERROR=$5

BA9D "PATH NOT FOUND" ERROR=$6,$7

BA99 "I/O ERROR"  ERROR=$8

BAA0 "DISK FULL" ERROR=$9

BAD2 "FILE LOCKED" ERROR=$A

BAD9 "INVALID PARAMETER" ERROR=$B

BAE3 "RAM TOO LARGE" ERROR=$C

BAF0 "FILE TYPE MISMATCH" ERROR=$D
### BASIC Interpreter (BI) -- V1.1 -- 18 JUN 84

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRPT</th>
<th>CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>BC92</td>
<td></td>
<td>LENGTH OF STRINGS ONLY</td>
</tr>
<tr>
<td>BC94</td>
<td></td>
<td>OPEN FILES' BUFFER MSB</td>
</tr>
<tr>
<td>BC98</td>
<td></td>
<td>OPEN EXEC FILE BUFFER MSB</td>
</tr>
<tr>
<td>BC9C</td>
<td></td>
<td>OPEN FILES' REFERENCE NUMBERS</td>
</tr>
<tr>
<td>BCA3</td>
<td></td>
<td>OPEN EXEC FILE REPNUM</td>
</tr>
<tr>
<td>BCA4</td>
<td></td>
<td>CURRENT RECORD LENGTH</td>
</tr>
<tr>
<td>BCA6</td>
<td></td>
<td>NOT USED</td>
</tr>
<tr>
<td>BCA9</td>
<td></td>
<td>CHARACTER TO FLUSH WHEN PARSING (BLANK)</td>
</tr>
<tr>
<td>BCA9</td>
<td></td>
<td>MAXIMUM LENGTH TO PARSE</td>
</tr>
<tr>
<td>BCA8</td>
<td></td>
<td>ADDRESS OF COMMAND HANDLING ROUTINE</td>
</tr>
<tr>
<td>BCA6</td>
<td></td>
<td>SIZE OF KEYWORD VALUE -1 IN BYTES</td>
</tr>
<tr>
<td>BCA4</td>
<td></td>
<td>OFFSET INTO KEYWORD PARMS TO LAST BYTE</td>
</tr>
<tr>
<td>BCAF</td>
<td></td>
<td>GENERAL PURPOSE 4 BYTE ACCUMULATOR</td>
</tr>
<tr>
<td>BCB3</td>
<td></td>
<td>MONTH</td>
</tr>
<tr>
<td>BCB4</td>
<td></td>
<td>DAY</td>
</tr>
<tr>
<td>BCB5</td>
<td></td>
<td>YEAR</td>
</tr>
<tr>
<td>BCB6</td>
<td></td>
<td>ERROR MSG LEN OR LINE LEN FOR CAT/CATALOG</td>
</tr>
<tr>
<td>BCB7</td>
<td></td>
<td>ENTRY LENGTH IN DIRECTORY FILE</td>
</tr>
<tr>
<td>BCBB</td>
<td></td>
<td>ENTRIES PER BLOCK IN DIRECTORY FILE</td>
</tr>
<tr>
<td>BCB9</td>
<td></td>
<td>FILE COUNT FROM DIRECTORY FILE</td>
</tr>
<tr>
<td>BCBB</td>
<td></td>
<td>DIRECTORY, ENTRY NUMBER COUNTER</td>
</tr>
<tr>
<td>BCBE</td>
<td></td>
<td>COMMAND OR PATH LENGTH</td>
</tr>
<tr>
<td>BCBD</td>
<td></td>
<td>TXBUF (COMMAND OR PATHNAME STRING)</td>
</tr>
<tr>
<td>BCFD</td>
<td></td>
<td>NOT USED</td>
</tr>
<tr>
<td>BCFE</td>
<td></td>
<td>OPEN FILE NAME TABLE (EACH ENTRY IS 32 BYTES LONG)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(THERE ARE 8 ENTRIES)</td>
</tr>
<tr>
<td>BCFE</td>
<td></td>
<td>FILE 0: LENGTH OF NAME</td>
</tr>
<tr>
<td>BCFH</td>
<td></td>
<td>FILE 0: L VALUE LSB</td>
</tr>
<tr>
<td>BD00</td>
<td></td>
<td>FILE 0: L VALUE MSB</td>
</tr>
<tr>
<td>BD01</td>
<td></td>
<td>FILE 0: START OF NAME STRING</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(FILE NAME IS STORED BACKWARDS)</td>
</tr>
<tr>
<td>BDFE</td>
<td></td>
<td>LAST 2 BYTES NOT USED</td>
</tr>
</tbody>
</table>

---

###UserCodeExtensions

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRPT</th>
<th>CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAPC</td>
<td></td>
<td>&quot;PROGRAM COMMAND&quot; ERROR=$E</td>
</tr>
<tr>
<td>BB07</td>
<td></td>
<td>&quot;NOT DIRECT&quot; ERROR=$F</td>
</tr>
<tr>
<td>BB11</td>
<td></td>
<td>&quot;SYNTAX ERROR&quot; ERROR=$10</td>
</tr>
<tr>
<td>BB19</td>
<td></td>
<td>&quot;DIRECT&quot; ERROR=$11</td>
</tr>
<tr>
<td>BB21</td>
<td></td>
<td>&quot;FILE NAME&quot; FILE NAME ERROR=$13</td>
</tr>
<tr>
<td>BB29</td>
<td></td>
<td>&quot;DUPLICATE&quot; ERROR=$14</td>
</tr>
<tr>
<td>BB34</td>
<td></td>
<td>&quot;FILE BUFFER STILL OPEN&quot; ERROR=$15</td>
</tr>
<tr>
<td>BB3B</td>
<td></td>
<td>&quot;FILE(S)&quot; VARIABLES ***************</td>
</tr>
<tr>
<td>BB40</td>
<td></td>
<td>VARIABLES ***************</td>
</tr>
<tr>
<td>BB47</td>
<td></td>
<td>Pages to Allocate/Free</td>
</tr>
<tr>
<td>BB47</td>
<td></td>
<td>NUMBER OF BUFFERS FOR GARBAGE COLLECTION</td>
</tr>
<tr>
<td>BB4B</td>
<td></td>
<td>NOT USED BUFFERS</td>
</tr>
<tr>
<td>BB4B</td>
<td></td>
<td>TOP OF BUFFER</td>
</tr>
<tr>
<td>BB4A</td>
<td></td>
<td>BOTTOM OF BUFFER</td>
</tr>
<tr>
<td>B84B-$B87A</td>
<td></td>
<td>NOT USED ***************</td>
</tr>
<tr>
<td>BB4B</td>
<td></td>
<td>NOT USED VARIABLES ***************</td>
</tr>
<tr>
<td>BC78</td>
<td></td>
<td>SAVED HUM - WORKSPACE SIZE</td>
</tr>
<tr>
<td>BC82</td>
<td></td>
<td>GARBAGE COLLECT MARKED GC: ****</td>
</tr>
<tr>
<td>BC82</td>
<td></td>
<td>AREA MSB</td>
</tr>
<tr>
<td>BB7C</td>
<td></td>
<td>GC: HIRABLE OF PAGES IN WORKAREA</td>
</tr>
<tr>
<td>BC7D</td>
<td></td>
<td>GC: WORK GE (START OF STRINGS TO COPY)</td>
</tr>
<tr>
<td>BC7E</td>
<td></td>
<td>GC: NUMBGE (END OF STRINGS TO COPY)</td>
</tr>
<tr>
<td>BC7F</td>
<td></td>
<td>GC: LARGEMB LSBB</td>
</tr>
<tr>
<td>BCBB</td>
<td></td>
<td>GC: HIRABLE MSB+1</td>
</tr>
<tr>
<td>BC10</td>
<td></td>
<td>ARRAYS SY OF STRING AREA (ALSO PGM START)</td>
</tr>
<tr>
<td>BC82</td>
<td></td>
<td>ARRAYS SY OF STRING AREA</td>
</tr>
<tr>
<td>BC83</td>
<td></td>
<td>GC: START FACTOR FOR STRING POINTERS</td>
</tr>
<tr>
<td>BC85</td>
<td></td>
<td>GC: END LING BLOCK BUFFER</td>
</tr>
<tr>
<td>BC87</td>
<td></td>
<td>MSB ADJU** STORED VARIABLES FILE HEADER ***</td>
</tr>
<tr>
<td>BC88</td>
<td></td>
<td>PAGE FILE LEN OF SIMPLE/ARRAY VARS</td>
</tr>
<tr>
<td>BC89</td>
<td></td>
<td>SIMPLE VARS ONLY</td>
</tr>
<tr>
<td>BC8B</td>
<td></td>
<td>VARS WERE COMBINED</td>
</tr>
<tr>
<td>BC8B</td>
<td></td>
<td>LEN OF SIMILAR COMBINED VARIABLES/STRINGS</td>
</tr>
<tr>
<td>BC90</td>
<td></td>
<td>** COMBINED VARIABLES/STRINGS</td>
</tr>
<tr>
<td>BC8E</td>
<td></td>
<td>POINTER</td>
</tr>
</tbody>
</table>
| BC90 |         | LENGTH OF STRING
**BASIC INTERPRETER GLOBAL PAGE**

This page of memory is rigidly defined by the ProDOS BI. Fields given here will not move in later versions of ProDOS and may be referenced by external, user-written programs. Future additions to the global page may be made in areas which are marked "Not used".

<table>
<thead>
<tr>
<th>ADDR</th>
<th>LABEL</th>
<th>CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>BE00-BE02</td>
<td>BI.ENTRY</td>
<td>JMP to WARMDO$ (BI warmstart vector).</td>
</tr>
<tr>
<td>BE03-BE05</td>
<td>DOSCMD</td>
<td>JMP to SYNTAX (BI command line parse and execute).</td>
</tr>
<tr>
<td>BE06-BE08</td>
<td>EXTRNMD</td>
<td>JMP to user-installed external command parser.</td>
</tr>
<tr>
<td>BE09-BE0B</td>
<td>ERROUT</td>
<td>JMP to BI error handler.</td>
</tr>
<tr>
<td>BE0C-BE0E</td>
<td>PRINTERR</td>
<td>JMP to BI error message print routine.</td>
</tr>
<tr>
<td>BE0F</td>
<td>ERRCODE</td>
<td>ProDOS error code (also at $DE, Applesoft ONERR code).</td>
</tr>
<tr>
<td>BE10-BE12</td>
<td>OUTVEC</td>
<td>Default output vector in monitor and for each slot (1-7).</td>
</tr>
<tr>
<td>BE13-BE15</td>
<td>INVEC</td>
<td>Default input vector in monitor for each slot (1-7).</td>
</tr>
<tr>
<td>BE16-BE18</td>
<td>VECTOUT</td>
<td>Current output vector.</td>
</tr>
<tr>
<td>BE19-BE1B</td>
<td>VECTIN</td>
<td>Current input vector.</td>
</tr>
<tr>
<td>BE1C-BE1E</td>
<td>VDOSIO</td>
<td>BI's output intercept address.</td>
</tr>
<tr>
<td>BE1F-BE23</td>
<td>BIN</td>
<td>BI's input intercept address.</td>
</tr>
<tr>
<td>BE24-BE26</td>
<td>VSYSIO</td>
<td>BI's internal redirection by STATE.</td>
</tr>
<tr>
<td>BE27-BE29</td>
<td>DEFSLT</td>
<td>Default slot.</td>
</tr>
<tr>
<td>BE2A-BE2C</td>
<td>DEFDRV</td>
<td>Default drive.</td>
</tr>
<tr>
<td>BE2D-BE2F</td>
<td>PREGA</td>
<td>A-register savearea.</td>
</tr>
<tr>
<td>BE30-BE32</td>
<td>PREGX</td>
<td>X-register savearea.</td>
</tr>
<tr>
<td>BE33-BE35</td>
<td>PREGY</td>
<td>Y-register savearea.</td>
</tr>
<tr>
<td>BE36-BE38</td>
<td>DTRACE</td>
<td>Applesoft TRACE is enabled flag (MSB on).</td>
</tr>
<tr>
<td>BE39-BE3B</td>
<td>STATE</td>
<td>Current intercept state. 0 = immediate command mode. &gt;0 = deferred.</td>
</tr>
<tr>
<td>BE3C-BE3E</td>
<td>EXACTV</td>
<td>EXEC file active flag (MSB on).</td>
</tr>
<tr>
<td>BE3F-BE41</td>
<td>IFILACTV</td>
<td>READ file active flag (MSB on).</td>
</tr>
<tr>
<td>BE42-BE44</td>
<td>OFILACTV</td>
<td>WRITE file active flag (MSB on).</td>
</tr>
<tr>
<td>BE45-BE47</td>
<td>PFXACTV</td>
<td>PREFIX read active flag (MSB on).</td>
</tr>
<tr>
<td>BE48-BE4A</td>
<td>DIRFLG</td>
<td>File being READ is a DIR file (MSB on).</td>
</tr>
<tr>
<td>BE4B-BE4D</td>
<td>EDIRFLG</td>
<td>End of directory flag (no longer used).</td>
</tr>
<tr>
<td>BE4E-BE50</td>
<td>STRINGS</td>
<td>String space count used to determine when to garbage collect.</td>
</tr>
<tr>
<td>BE51-BE53</td>
<td>BUFSIZE</td>
<td>Buffered WRITE data length.</td>
</tr>
<tr>
<td>BE54-BE56</td>
<td>INPTR</td>
<td>Command line assembly length.</td>
</tr>
<tr>
<td>BE57-BE59</td>
<td>CHRLAST</td>
<td>Previous output character (for recursion check).</td>
</tr>
<tr>
<td>BE5A-BE5C</td>
<td>OPENCNT</td>
<td>Number of files open (not counting EXEC).</td>
</tr>
<tr>
<td>BE5D-BE5F</td>
<td>YXFILE</td>
<td>EXEC file being closed flag (MSB on).</td>
</tr>
<tr>
<td>BE60-BE62</td>
<td>CATFLAG</td>
<td>Line type to format next in DIR file READ.</td>
</tr>
<tr>
<td>BE63-BE65</td>
<td>XPNNADDR</td>
<td>External command handler address.</td>
</tr>
<tr>
<td>BE66-BE68</td>
<td>XLEN</td>
<td>Length of command name (less one).</td>
</tr>
</tbody>
</table>
### ProDOS BI Global Page

<table>
<thead>
<tr>
<th>ADDR</th>
<th>LABEL</th>
<th>CONTENTS</th>
</tr>
</thead>
</table>
| BE53 | XCNUM | Number of command:

- `$0 = external`
- `$9B = OPEN`
- `$15 = WRITE`
- `$91 = IN#`
- `$9C = READ`
- `$16 = APPEND`
- `$92 = PR#`
- `$9D = SAVE`
- `$17 = CREATE`
- `$93 = CAT`
- `$9E = BLOAD`
- `$18 = DELETE`
- `$94 = PRE`
- `$9F = BSAVE`
- `$19 = PREFIX`
- `$95 = BYE`
- `$A0 = CHAIN`
- `$1A = RENAME`
- `$96 = RUN`
- `$11 = CLOSE`
- `$1B = UNLOCK`
- `$97 = BRUN`
- `$12 = FLUSH`
- `$1C = VERIFY`
- `$98 = EXEC`
- `$13 = NOMON`
- `$1D = CATALOG`
- `$99 = LOAD`
- `$14 = STORE`
- `$1E = RESTORE`
- `$9A = LOCK`
- `$1F = POSITION`

### ProDOS BI Global Page

<table>
<thead>
<tr>
<th>ADDR</th>
<th>LABEL</th>
<th>CONTENTS</th>
</tr>
</thead>
</table>
| BE54-BE55 | FBITS | 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 permitted.
- `$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`: S or D: slot/drive permitted.
- `$4000`: F: field permitted.

(V always permitted but ignored.)

### ProDOS BI Global Page

<table>
<thead>
<tr>
<th>ADDR</th>
<th>LABEL</th>
<th>CONTENTS</th>
</tr>
</thead>
</table>
| BE56-BE57 | FBITS | Operands found on command line. Same bit assignments as above.

BE58-BE59

<table>
<thead>
<tr>
<th>ADDR</th>
<th>LABEL</th>
<th>CONTENTS</th>
</tr>
</thead>
</table>
| BE58-BE59 | VADDR | A keyword value.
| BE5A-BE5C | VBYTE | B keyword value.
| BE5D-BE5E | VENDA | E keyword value.
| BE5F-BE60 | VLNTH | L keyword value.
| BE61 | VSLOT | S keyword value.
| BE62 | VDRIV | D keyword value.
| BE63-BE64 | VFELD | F keyword value.
| BE65-BE66 | VRDCD | R keyword value.
| BE67 | VVOLM | V keyword value (ignored).

BE68-BE69

<table>
<thead>
<tr>
<th>ADDR</th>
<th>LABEL</th>
<th>CONTENTS</th>
</tr>
</thead>
</table>
| BE68-BE69 | VLINE | @ keyword value.
| BE6A | VTYPE | T keyword value (in hex).
| BE6B | VIOSLT | PR# or IN# slot number value.

BE6C-BE6D

<table>
<thead>
<tr>
<th>ADDR</th>
<th>LABEL</th>
<th>CONTENTS</th>
</tr>
</thead>
</table>
| BE6C-BE6D | VPATH1 | Primary pathname buffer (address of length byte).
| BE6E-BE6F | VPATH2 | Secondary pathname buffer (address of length byte).

BE70-BE74

<table>
<thead>
<tr>
<th>ADDR</th>
<th>LABEL</th>
<th>CONTENTS</th>
</tr>
</thead>
</table>
| BE70-BE74 | GOSYSTEM | Call the MLI using the parameter tables which follow.

BE75

<table>
<thead>
<tr>
<th>ADDR</th>
<th>LABEL</th>
<th>CONTENTS</th>
</tr>
</thead>
</table>
| BE75 | SYSCALL | MLI call number for this call.

BE86-BE87

<table>
<thead>
<tr>
<th>ADDR</th>
<th>LABEL</th>
<th>CONTENTS</th>
</tr>
</thead>
</table>
| BE86-BE87 | SYSPARM | Address of MLI parameter list for this call.

BE88-BE89

<table>
<thead>
<tr>
<th>ADDR</th>
<th>LABEL</th>
<th>CONTENTS</th>
</tr>
</thead>
</table>
| BE88-BE89 | BADCALL | Return from MLI call.

BE8B-BE8E

<table>
<thead>
<tr>
<th>ADDR</th>
<th>LABEL</th>
<th>CONTENTS</th>
</tr>
</thead>
</table>
| BE8B-BE8E | BADCALL | MLI error return: translate error code to BI error number.

BE9F

<table>
<thead>
<tr>
<th>ADDR</th>
<th>LABEL</th>
<th>CONTENTS</th>
</tr>
</thead>
</table>
| BE9F | BISPARSE | Not used.

BEA0-BEAB

<table>
<thead>
<tr>
<th>ADDR</th>
<th>LABEL</th>
<th>CONTENTS</th>
</tr>
</thead>
</table>
| BEA0-BEAB | SCREATE | CREATE parameter list.
| BEAC-BEAE | SCGPRFX | GET_PREFIX, SET_PREFIX, DESTROY parameter list.

BEAF-BEB3

<table>
<thead>
<tr>
<th>ADDR</th>
<th>LABEL</th>
<th>CONTENTS</th>
</tr>
</thead>
</table>
| BEAF-BEB3 | SRENAM | RENAME parameter list.
| BEB4-BEC5 | SETFILEINFO | GET_FILE_INFO, SET_FILE_INFO parameter list.

BEC6-BEC7

<table>
<thead>
<tr>
<th>ADDR</th>
<th>LABEL</th>
<th>CONTENTS</th>
</tr>
</thead>
</table>
| BEC6-BEC7 | SONLINE | ONLINE, SET_MARK, GET_MARK, SET_EOF, GET_EOF, SET_BUF, GET_BUF, QUIT parameter list.

BEC8-BED0

<table>
<thead>
<tr>
<th>ADDR</th>
<th>LABEL</th>
<th>CONTENTS</th>
</tr>
</thead>
</table>
| BEC8-BED0 | SOPEN | OPEN parameter list.
| BED1-BED4 | SNNEWLN | SET_NEWLINE parameter list.
| BED5-BEDC | SREAD | READ, WRITE parameter list.
| BEDD-BEE6 | SCLOSE | CLOSE, FLUSH parameter list.

BEEF-BEF7

<table>
<thead>
<tr>
<th>ADDR</th>
<th>LABEL</th>
<th>CONTENTS</th>
</tr>
</thead>
</table>
| BEEF-BEF7 | GETBUFFER | GETBUFFER buffer allocation subroutine vector.

BEE8-BEF9

<table>
<thead>
<tr>
<th>ADDR</th>
<th>LABEL</th>
<th>CONTENTS</th>
</tr>
</thead>
</table>
| BEE8-BEF9 | FREEBUFFER | FREEBUFFER buffer free subroutine vector.

BEFC-BEFF

<table>
<thead>
<tr>
<th>ADDR</th>
<th>LABEL</th>
<th>CONTENTS</th>
</tr>
</thead>
</table>
| BEFC-BEFF | HINEM | Original HINEM MSB.

Not used.
Disk Controller Boot ROM -- Apple II/IIE

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION</th>
<th>CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>C66E</td>
<td>NO, SEE IF i</td>
<td>CONTENTS</td>
</tr>
<tr>
<td>C670</td>
<td>YES, DELAY i</td>
<td>CONTENTS</td>
</tr>
<tr>
<td>C671</td>
<td>CHECK REGISTER i</td>
<td>CONTENTS</td>
</tr>
<tr>
<td>C674</td>
<td>LOOP UNTIL REGISTER TO CLEAR</td>
<td>CONTENTS</td>
</tr>
<tr>
<td>C676</td>
<td>IS IT A $96</td>
<td>CONTENTS</td>
</tr>
<tr>
<td>C678</td>
<td>YES, WE FOUND ONE PREVIOUSLY?</td>
<td>CONTENTS</td>
</tr>
<tr>
<td>C67A</td>
<td>NO, HAVE WE AN ADDRESS HEADER</td>
<td>CONTENTS</td>
</tr>
<tr>
<td>C67B</td>
<td>IF NOT, START OVER</td>
<td>CONTENTS</td>
</tr>
<tr>
<td>C683</td>
<td>LOAD ADDRESS FIELD</td>
<td>CONTENTS</td>
</tr>
</tbody>
</table>

C683 **DETERMINE IF THERE IS MORE TO DO**

C685 **SAVE VALUE AT CURSOR**

C687 **READ DATA REPEATED WILL BE TRACK ON LAST PASS**

C68A **LOOP UNTIL REGISTER**

C68C **SHIFT BITS**

C68D **SAVE FOR LATER**

C68F **READ REGISTER**

C692 **LOOP UNTIL**

C694 **COMBINE WITH**

C696 **DECREMENT COUNT**

C697 **NO, DO ANOTHER**

C699 **KEEP THE STALE**

C6A8 **IS THIS SECTION CLEAN**

C6A9 **NO, START OVER**

C6A9 **GET TRACK FOR**

C6A1 **NO, START OVER**

C6A4 **YES, INDICATE**

C666 **DATA FIELD**

C6A8 **MAKE auxillary buffer**

C6A9 **READ DATA REGISTER**

C6AF **EXCLUSIVE-OR**

C6B4 **DECREMENT COUNT**

C6B5 **STORE BYTE**

C6B8 **LOOP UNTIL**

C6BA **INITIALIZE**

C6BC **READ DATA REGISTER**

C6BF **LOOP UNTIL**

C6C1 **EXCLUSIVE-OR**

C6C6 **STORE BYTE**

C6CB **INCREMENT OFFSET**

C6CB **READ DATA REGISTER**

C6CE **LOOP UNTIL VALID**

C6D0 **IS CHECKSUM OKAY**

C6D2 **START OVER**

C6D5 **MERGE MAIN AND AUXILIARY BUFFERS**

C6D7 **INITIALIZE OFFSET**

C6D9 **DECREMENT OFFSET**

C6DA **IF LESS THAN ZERO RESET IT**

C6DB **GET BYTE FROM MAIN BUFFER**

C6E1 **ROLL IN TWO BITS FROM AUXILIARY BUFFER**

C6E6 **SAVE COMPLETED DATA BYTE**

C6E8 **INCREMENT OFFSET**

C6E9 **LOOP UNTIL WHOLE BUFFER IS DONE**

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

C6ED **INCREMENT MAIN BUFFER POINTER**

C6F1 **IS THERE ANOTHER SECTOR TO LOAD**

C6FB **5 BYTES AT END OF PAGE ARE UNUSED**
C522

**Module Starting Address**

* BOOT ROM - APPLE //C CONTROLLER ROM
* THIS CODE RESIDES FROM $C552 TO $C6FF. IT LOADS TRACK 0 SECTOR 0 INTO RAM AT $800 AND JUMPS TO IT. IF BOOT FAILS IT THEN TRIES TO BOOT SLOT 5, THE PROTOCOL CONVERTER.
* THIS IS THE VERSION OF THE //C ROM THAT SUPPORTS THE UNIDISK 3.5, 26 JULY 85.

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

&001 SLOT PAGE PUT HERE DURING AUTOBOOT
&003 RETRY COUNT (HIGH BYTE)
&026 SECTOR BUFFER POINTER
&028 SLOT NUMBER * 16 FOR INDEX
&03C WORD BYTE
&03D SECTOR WANTED
&040 TRACK FOUND
&041 TRACK WANTED
&04F DRIVE TO BOOT FROM

********** External Addresses **********

&030 AUXILIARY BUFFER
&036 TRANSLATE TABLE
&07B SCREEN LOCATION
&080 SECTORS TO LOAD
&081 ENTRY POINT
&082 PHASE OFF
&083 PHASE ON
&088 MOTOR OFF
&089 MOTOR ON
&08C READ DATA REGISTER
&08E SET READ MODE
&09A DRIVE SELECT
&0A8 MONITOR WAIT ROUTINE

********** Slot 5 Code **********

THE FOLLOWING TWO ROUTINES ARE IN THE $C560 AREA BUT ARE USED BY THE $C600 LOGIC.

C522

********** Bootfail **********

COME HERE IF BOOT FAILS. PUT MESSAGE ON SCREEN AND GO TO SLEEP FOREVER.

C552

17 CHARACTERS IN MESSAGE
C557 PUT AT BOTTOM OF SCREEN ($7DB)
C55D THEN GO TO SLEEP >C55D

C55F 'Check Disk Drive'

C56F ********** Skip over Miscellaneous Code **********

C56F SLOT 5 LOGIC IN HERE

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

C5BE 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?
C598 NO, TRY ANOTHER PATTERN >C5AA
C59A YES, TURN OFF RIGHTMOST OF EACH GROUP OF ZEROS
C59C FLIP BITS, PAIR OF ZERO BITS NOW SINGLE BIT, ETC
C59E HIGH BIT ALWAYS ON/TURN OFF BIT WE MISSED BEFORE
C5A0 >C5AA
C5A2 SHIFT PATTERN RIGHT, MUST HAVE ONLY ONE BIT ON
C5A3 IF MORE THAN ONE BIT ON, TRY ANOTHER PATTERN >C5AA
C5A5 FOUND ONE, GET TABLE VALUE
C5A6 AND STORE IT IN TABLE ($536)
C5A9 INCREMENT TABLE VALUE INDICATOR
C5AA GET NEXT BIT PATTERN, DONE YET?
C5AB 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
<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
<th>NEXT OBJECT ADDR</th>
</tr>
</thead>
<tbody>
<tr>
<td>C5F5</td>
<td>*************** JUMP TO BOOT FAIL</td>
<td>C5F6</td>
</tr>
<tr>
<td>C5F5</td>
<td>BRANCH TO BOOTFAIL</td>
<td>C552</td>
</tr>
<tr>
<td>C5F8</td>
<td>REMAINING 8 BYTES NOT USED BY DISK II</td>
<td>C576</td>
</tr>
<tr>
<td>C680</td>
<td>*************** INITIALIZATION</td>
<td></td>
</tr>
<tr>
<td>C680</td>
<td>SIGNATURE</td>
<td></td>
</tr>
<tr>
<td>C682</td>
<td>SET DRIVE -&gt; 1</td>
<td></td>
</tr>
<tr>
<td>C684</td>
<td>INITIALIZE RETRY COUNT (HIGH BYTE)</td>
<td></td>
</tr>
<tr>
<td>C688</td>
<td>*************** SELECT DRIVE</td>
<td></td>
</tr>
<tr>
<td>C688</td>
<td>*** AND TURN IT ON ***************</td>
<td></td>
</tr>
<tr>
<td>C68E</td>
<td>****</td>
<td></td>
</tr>
<tr>
<td>C68F</td>
<td>INITIALIZE SLOT (6)</td>
<td></td>
</tr>
<tr>
<td>C69D</td>
<td>INITIALIZE DEVICE (1)</td>
<td></td>
</tr>
<tr>
<td>C69F</td>
<td>SAVE DRIVE NUMBER ON OR 2)</td>
<td></td>
</tr>
<tr>
<td>C6A0</td>
<td>INCREASE READ MODE (C, STACK</td>
<td></td>
</tr>
<tr>
<td>C6A2</td>
<td>GET DRIVE NUMBER BACK</td>
<td></td>
</tr>
<tr>
<td>C6A3</td>
<td>SELECT APPROPRIATE B2</td>
<td></td>
</tr>
<tr>
<td>C6A4</td>
<td>TURN MOTOR ON (C0B8 DRIVE (C0EA)</td>
<td></td>
</tr>
<tr>
<td>C61D</td>
<td>*************** RECALIBRATE</td>
<td></td>
</tr>
<tr>
<td>C61D</td>
<td>PREPARE TO STEP THE</td>
<td></td>
</tr>
<tr>
<td>C61E</td>
<td>DISK ARM ***************</td>
<td></td>
</tr>
<tr>
<td>C61F</td>
<td>TUR</td>
<td></td>
</tr>
<tr>
<td>C622</td>
<td>PUT COUNTER IN A REG (4)</td>
<td></td>
</tr>
<tr>
<td>C623</td>
<td>CREATE A PHASE NUMBERST</td>
<td></td>
</tr>
<tr>
<td>C625</td>
<td>DOUBLE IT FOR PROPER (0-3)</td>
<td></td>
</tr>
<tr>
<td>C626</td>
<td>COMBINE WITH SLOT EFG INDEX</td>
<td></td>
</tr>
<tr>
<td>C627</td>
<td>PUT INDEX IN X REGISTER FINAL INDEX</td>
<td></td>
</tr>
<tr>
<td>C627</td>
<td>TUR</td>
<td></td>
</tr>
<tr>
<td>C629</td>
<td>DELAY ABOUT 20 MICROSECONDS</td>
<td></td>
</tr>
<tr>
<td>C631</td>
<td>DEC</td>
<td></td>
</tr>
<tr>
<td>C632</td>
<td>LOOP UNTIL ALL 80 AND DONE -&gt; C61F</td>
<td></td>
</tr>
<tr>
<td>C634</td>
<td>*************** INITIALIZATION</td>
<td></td>
</tr>
<tr>
<td>C634</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C636</td>
<td>SECTOR TO FIND -&gt; &amp;H</td>
<td></td>
</tr>
<tr>
<td>C638</td>
<td>TRACK TO FIND -&gt; &amp;H</td>
<td></td>
</tr>
<tr>
<td>C63A</td>
<td>BUILD THE TRANSLATE TABLE &lt;C58E&gt;</td>
<td></td>
</tr>
<tr>
<td>C63D</td>
<td>*************** COUNT RETRY COUNTERS AND INDICATE ERROR IF BOOT FAILS********</td>
<td></td>
</tr>
<tr>
<td>C63F</td>
<td>CLEAR THE CARRY UNIT</td>
<td></td>
</tr>
<tr>
<td>C640</td>
<td>PUSH STATUS ON STACK</td>
<td></td>
</tr>
<tr>
<td>C641</td>
<td>KEEP STACK CLEAN</td>
<td></td>
</tr>
<tr>
<td>C642</td>
<td>GET SLOT</td>
<td></td>
</tr>
<tr>
<td>C644</td>
<td>DECREMENT RETRY COUNT, TRY AGAIN?</td>
<td>C656</td>
</tr>
<tr>
<td>C646</td>
<td>YES, GO DO IT &gt;&gt;C656</td>
<td></td>
</tr>
<tr>
<td>C648</td>
<td>NO, TURN DRIVE OFF (C0B8)</td>
<td>C64B</td>
</tr>
<tr>
<td>C64B</td>
<td>AUTO BOOT FROM SLOT 67</td>
<td>C64F</td>
</tr>
<tr>
<td>C650</td>
<td>NO, FAIL NOW &gt;&gt;C5F5</td>
<td>C576</td>
</tr>
<tr>
<td>C651</td>
<td>MAYBE SLOT 5 WILL TALK TO US &gt;&gt;C500</td>
<td></td>
</tr>
<tr>
<td>C654</td>
<td>TWO BITES NOT USED &gt;&gt;B002</td>
<td></td>
</tr>
<tr>
<td>C656</td>
<td>&gt;&gt;</td>
<td></td>
</tr>
<tr>
<td>C657</td>
<td>DECREMENT RETRY COUNT (LOW BYTE)</td>
<td></td>
</tr>
<tr>
<td>C658</td>
<td>IF NOT ZERO, TRY AGAIN &gt;&gt;C5E</td>
<td></td>
</tr>
<tr>
<td>C65A</td>
<td>IF SO, GO DECREMENT RETRY COUNT (HIGH BYTE) &gt;&gt;C641</td>
<td></td>
</tr>
<tr>
<td>C65C</td>
<td>SPACE FILLER TO POSITION CODE BELOW &gt;&gt;C63D</td>
<td></td>
</tr>
<tr>
<td>C65E</td>
<td>*************** SEARCH FOR A VALID HEADER ***************</td>
<td></td>
</tr>
<tr>
<td>C65E</td>
<td>CHECK DATA REGISTER (C0B8)</td>
<td></td>
</tr>
<tr>
<td>C660</td>
<td>LOOP UNTIL DATA IS VALID &gt;&gt;C65E</td>
<td></td>
</tr>
<tr>
<td>C662</td>
<td>IS IT A $57?</td>
<td></td>
</tr>
<tr>
<td>C665</td>
<td>NO, TRY AGAIN &gt;&gt;C657</td>
<td></td>
</tr>
<tr>
<td>C667</td>
<td>SELECT ADDRESS REGISTER AGAIN (C0B8)</td>
<td></td>
</tr>
<tr>
<td>C66A</td>
<td>LOOP UNTIL VALID &gt;&gt;C667</td>
<td></td>
</tr>
<tr>
<td>C66C</td>
<td>IS IT AN $AA</td>
<td></td>
</tr>
<tr>
<td>C66E</td>
<td>NO, SEE IF ITS A $D5 &gt;&gt;C663</td>
<td></td>
</tr>
<tr>
<td>C670</td>
<td>YES, DELAY FOR REGISTER TO CLEAR</td>
<td></td>
</tr>
<tr>
<td>C671</td>
<td>CHECK REGISTER (C0B8)</td>
<td></td>
</tr>
<tr>
<td>C674</td>
<td>LOOP UNTIL VALID &gt;&gt;C671</td>
<td></td>
</tr>
<tr>
<td>C676</td>
<td>IS IT A $63</td>
<td></td>
</tr>
<tr>
<td>C678</td>
<td>YES, WE FOUND AN ADDRESS HEADER &gt;&gt;C683</td>
<td></td>
</tr>
<tr>
<td>C67A</td>
<td>NO, HAVE WE FOUND ONE PREVIOUSLY?</td>
<td></td>
</tr>
<tr>
<td>C67B</td>
<td>IF NOT, START OVER &gt;&gt;C63F</td>
<td></td>
</tr>
<tr>
<td>C67D</td>
<td>WAS IT AN $AD?</td>
<td></td>
</tr>
<tr>
<td>C67F</td>
<td>YES, WE FOUND A DATA HEADER &gt;&gt;C6A6</td>
<td></td>
</tr>
<tr>
<td>C681</td>
<td>NO, START OVER &gt;&gt;C63F</td>
<td></td>
</tr>
<tr>
<td>C683</td>
<td>*************** DECODE ADDRESS FIELD ***************</td>
<td></td>
</tr>
<tr>
<td>C683</td>
<td>INITIALIZE COUNTER</td>
<td></td>
</tr>
<tr>
<td>C684</td>
<td>SAVE VALUE DECODED, WILL BE TRACK ON LAST PASS</td>
<td></td>
</tr>
<tr>
<td>C687</td>
<td>READ DATA REGISTER (C0B8)</td>
<td></td>
</tr>
<tr>
<td>C68A</td>
<td>LOOP UNTIL DATA VALID &gt;&gt;C687</td>
<td></td>
</tr>
<tr>
<td>C68C</td>
<td>SHIFT BITS INTO POSITION 011011</td>
<td></td>
</tr>
<tr>
<td>C696</td>
<td>SAVE FOR LATER</td>
<td></td>
</tr>
<tr>
<td>C68F</td>
<td>READ REGISTER FOR NEXT BYTE (C0B8)</td>
<td></td>
</tr>
<tr>
<td>C692</td>
<td>LOOP UNTIL VALID &gt;&gt;C69F</td>
<td></td>
</tr>
<tr>
<td>C694</td>
<td>COMBINE WITH PREVIOUS 011011 AND 011011</td>
<td></td>
</tr>
<tr>
<td>C696</td>
<td>DECREMENT COUNTER, DONE YET?</td>
<td></td>
</tr>
<tr>
<td>C697</td>
<td>NO, DO ANOTHER &gt;&gt;C685</td>
<td></td>
</tr>
<tr>
<td>C699</td>
<td>KEEP THE STACK CLEAN</td>
<td></td>
</tr>
<tr>
<td>C69A</td>
<td>IS THIS SECTOR WE WANT?</td>
<td></td>
</tr>
<tr>
<td>C69C</td>
<td>NO, START OVER &gt;&gt;C63F</td>
<td></td>
</tr>
<tr>
<td>C69E</td>
<td>GET TRACK FOUND</td>
<td></td>
</tr>
</tbody>
</table>
C6B5 DECREMENT OFFSET
C6B5 STORE BYTE IN AUXILIARY BUFFER  (0:02D6)
C6B6 LOOP UNTIL BUFFER FULL  >>C6A8
C6B8 INITIALIZE OFFSET (MAIN BUFFER)  1:0000
C6B9 READ DATA REGISTER (C8BC)
C6B9 LOOP UNTIL VALID  >>C6BC
C6C1 EXCLUSIVE-OR WITH TRANSLATE TABLE
C6C2 INCREMENT OFFSET  (02D6)
C6C2 STORE BYTE IN MAIN BUFFER
C6C3 IS CHECKSUM OKAY?  (02D6)
C6C4 NO, START OVER  >>C6A2
C6C5 ************ MERGE MAIN AND AUXILIARY BUFFERS************
C6C5 INITIALIZE OFFSET (MAIN BUFFER)
C6C7 INITIALIZE OFFSET (AUXILIARY BUFFER)
C6C9 DECREMENT OFFSET (AUX BUFFER)
C6D0 IF LESS THAN ZERO RESET IT  >>C6D7
C6D2 GET BYTE FROM MAIN BUFFER
C6D4 ROLL IN TWO BITS FROM AUXILIARY BUFFER
C6D6 SAVE COMPLETED DATA BYTE
C6D8 INCREMENT OFFSET (MAIN BUFFER)
C6D9 LOOP UNTIL WHOLE BUFFER IS DONE
C6D9 ************ DETERMINE IF THERE IS MORE ************
C6EB INCREMENT MAIN BUFFER POINTER
C6ED INCREMENT SECTOR NUMBER
C6F0 IS THERE ANOTHER SECTOR TO LOAD?  (Y, N)
C6F2 YES, GO DO IT  >>C6D3
C6F4 NO, ENTER CODE WE JUST LOADED  >>0008
C6FB 5 ZERO BYTES AT END OF PAGE

---

**Beneath Apple ProDOS Supplement**

---

**Disk Controller Boot ROM -- Apple IIc**

**ADDR** | **DESCRIPTION/CONTENTS** | **NEX**
--- | --- | ---
C6A8 | IS IT TRACK WE WANT? |
C6A2 | NO, START OVER  >>C63F |
C6A4 | YES, INDICATE ADDRESS FOUND, GO LOC |

C6A6 ************ READ DATA FIELD ************

**C6A6** | **DESCRIPTION/CONTENTS** | **ASK FOR DAS**
--- | --- | ---
C6A6 | INITIALIZE OFFSET (AUXILIARY BUFFER) |
C6A8 | READ DATA REGISTER (C8BC) |
C6A9 | LOOP UNTIL VALID  >>C6A8 |
C6AF | EXCLUSIVE-OR WITH TRANSLATE TABLE |
Disk II Boot ROM -- Apple IIgs

Address                Description/Contents

C600 MODULE 8

STARTING ADDRESS

C600

IF CM - APPLE IIgs

then Apple IIgs has slot 6

when this code is executed from a boot is attempted

IIGS E

ROM VERSION $0

C600

********** 20K PAGE ADDRESSES *******************

1 SLOT PACK

3 RETRY C65 PUT HERE DURING AUTOBOOT

26 SECTOR COUNT (HIGH BYTE)

2B SLOT NUMBER POINTER

3C WORKBYTE x16

3D SECTOR x16

40 TRACK POINTED

41 TRACK WRITE

4F DRIVE TESTED

C600

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

0300 AUXILIARY

0560 TRANSLATION BUFFER

07FE SLOTS THAT TABLE

07FE UTILITY THAT OWN C800-CFFF

0800 SECTORS BYTE FOR SLOT 6

0801 ENTRY PT. TO LOAD

C800 PHASED PLINT

C801 PHASED OFF

C808 MOTOR ON

C809 MOTOR OFF

C80C READ DATA

C80E SET REAL ADDRESS

C90A DRIVE ID MODE

E000 APPLE SELECT

FABA MONITOR: BASIC ENTRY POINT

FCA8 MONITOR: DISK CONTROLLER SEARCH LOOP

Wait Routine

C600

********** IIGS SOFT SWITCHES *******************

C625 SLTRONSEL, SLOT ROM CONFIGURATION BYTE

C635 SHADOW, ENABLES/DISABLES SHADOWING

C636 STATEREG. ONE BYTE SETS 8 SOFT SWITCHES

C600

********** C600 ENTRY POINT *******************

C000

SIG

C012 SET DRIVE TO DRIVE 1

C014 INITIALIZE RETRY COUNT (HIGH BYTE)

C015 PFS ON STACK

C01A SET NORMAL ZERO PAGE

C01C SET DATA BANK SAME AS PROGRAM BANK

C01F MAKE SLOT 6 CURRENT (07FE)

C62 B-BIT MEMORY AND INDEX OPERATIONS

C616 STORE P-REGISTER IN SCREEN HOLD (07FE)

C619 DISABLE INTERRUPTS

C61A GO DO SOME TASKS ELSEWHERE IN ROM <FF5500>

C61E RESTORE SLOT*16 TO X-REGISTER

C620 IF CARRY SET, I/O ERROR >>C62E

C62

COUNT RETRIES AND INDICATE ERROR **********************

IF BOOT FAILS

C622 INITIALIZE RETRY COUNT

C624 DISABLE INTERRUPTS (AGAIN)

C625 CLEAR CARRY AND

C626 PUT IT ON THE STACK.

C627 GET SLOT 16

C628 GET SLOT*16 IN X-REGISTER

C62A DECREMENT RETRY COUNT. TRY AGAIN?

C62C YES, GO DO IT >>C656

C62E NO, TURN DRIVE OFF (C088)

C631 CALL BOOT ERROR HANDLER <FF5971>

C635 ENABLE INTERRUPTS IF OK TO <C64B>

C638 AUTOBOOT FROM SLOT 67

C63D NO, GO TO BASIC >>C642

C63F YES, RETURN TO SLOT SEARCH LOOP >>FABA

C642 JUMP TO BASIC >>C600

C64

********** SUBROUTINE TO ENABLE INTERRUPTS *******************

C645 ALLOW SHADOWING AND I/O (C635)

C648 CHECK ORIGINAL P-REGISTER (07FE)

C64B INTERRUPT HIT HIGH?

C64D YES, LEAVE INTERRUPTS DISABLED >>C650

C65F NO, ENABLE INTERRUPTS

C650 RESTORE SECTOR TO ACCUMULATOR

C652 RETURN
<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>C653</strong></td>
<td>THREE BYTES NOT USED</td>
</tr>
<tr>
<td><strong>C655</strong></td>
<td>INCREMENT RETRIES</td>
</tr>
<tr>
<td><strong>C656</strong></td>
<td>DATA REGISTER (C00C)</td>
</tr>
<tr>
<td><strong>C657</strong></td>
<td>LOOP UNTIL DATA IS VALID &gt;&gt;C65E</td>
</tr>
<tr>
<td><strong>C658</strong></td>
<td>IS IT A $D57?</td>
</tr>
<tr>
<td><strong>C659</strong></td>
<td>NO, TRY AGAIN &gt;&gt;C657</td>
</tr>
<tr>
<td><strong>C660</strong></td>
<td>YES, CHECK REGISTER AGAIN (C00C)</td>
</tr>
<tr>
<td><strong>C661</strong></td>
<td>LOOP UNTIL VALID &gt;&gt;C667</td>
</tr>
<tr>
<td><strong>C662</strong></td>
<td>IS IT AN $A9?</td>
</tr>
<tr>
<td><strong>C663</strong></td>
<td>NO, USE IT'S $A5D &gt;&gt;C663</td>
</tr>
<tr>
<td><strong>C665</strong></td>
<td>YES, DELAY FOR REGISTER TO CLEAR</td>
</tr>
<tr>
<td><strong>C666</strong></td>
<td>YES, REGISTER (C00C)</td>
</tr>
<tr>
<td><strong>C669</strong></td>
<td>LOOP UNTIL VALID &gt;&gt;C671</td>
</tr>
<tr>
<td><strong>C670</strong></td>
<td>IS IT A $967?</td>
</tr>
<tr>
<td><strong>C671</strong></td>
<td>YES, WE FOUND AN ADDRESS HEADER &gt;&gt;C683</td>
</tr>
<tr>
<td><strong>C672</strong></td>
<td>NO, START OVER &gt;&gt;C625</td>
</tr>
<tr>
<td><strong>C673</strong></td>
<td>IS IT AN $A0D?</td>
</tr>
<tr>
<td><strong>C674</strong></td>
<td>YES, WE FOUND A DATA HEADER &gt;&gt;C66</td>
</tr>
<tr>
<td><strong>C675</strong></td>
<td>NO, START OVER &gt;&gt;C625</td>
</tr>
<tr>
<td><strong>C683</strong></td>
<td>DECODE ADDRESS FIELD</td>
</tr>
<tr>
<td><strong>C685</strong></td>
<td>LIZE COUNTER</td>
</tr>
<tr>
<td><strong>C687</strong></td>
<td>VALUE DECODED, WILL BE TRACK ON LAST PASS</td>
</tr>
<tr>
<td><strong>C687</strong></td>
<td>DATA REGISTER (C00C)</td>
</tr>
<tr>
<td><strong>C689</strong></td>
<td>LOOP UNTIL DATA VALID &gt;&gt;C687</td>
</tr>
<tr>
<td><strong>C68B</strong></td>
<td>SHIFT FOR LATER</td>
</tr>
<tr>
<td><strong>C68D</strong></td>
<td>SAVE FOR LATER</td>
</tr>
<tr>
<td><strong>C68F</strong></td>
<td>REGISTER FOR NEXT BYTE (C00C)</td>
</tr>
<tr>
<td><strong>C692</strong></td>
<td>LOOP UNTIL VALID &gt;&gt;C68F</td>
</tr>
<tr>
<td><strong>C694</strong></td>
<td>COMPARE WITH PREVIOUS 1X11X1X1 AND 1X11X1X1</td>
</tr>
<tr>
<td><strong>C696</strong></td>
<td>DECREMENT COUNTER. DONE YET?</td>
</tr>
<tr>
<td><strong>C697</strong></td>
<td>NO, DO ANOTHER &gt;&gt;C685</td>
</tr>
<tr>
<td><strong>C699</strong></td>
<td>STACK CLEAN</td>
</tr>
<tr>
<td><strong>C69A</strong></td>
<td>IS THIS SECTOR WE WANT?</td>
</tr>
<tr>
<td><strong>C69C</strong></td>
<td>NO, START OVER &gt;&gt;C625</td>
</tr>
<tr>
<td><strong>C69E</strong></td>
<td>GOTO TRACK FOUND</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>C6A6</strong></td>
<td>IS IT TRACK WE WANT?</td>
</tr>
<tr>
<td><strong>C6A8</strong></td>
<td>NO, START OVER &gt;&gt;C625</td>
</tr>
<tr>
<td><strong>C6A9</strong></td>
<td>YES, INDICATE ADDR FOUND, GO LOOK FOR DATA FIELD &gt;&gt;C628</td>
</tr>
<tr>
<td><strong>C6A6</strong></td>
<td>INITIALIZE OFFSET (AUXILIARY BUFFER)</td>
</tr>
<tr>
<td><strong>C6A8</strong></td>
<td>READ DATA REGISTER (C00C)</td>
</tr>
<tr>
<td><strong>C6A9</strong></td>
<td>LOOP UNTIL VALID &gt;&gt;C6A</td>
</tr>
<tr>
<td><strong>C6A8</strong></td>
<td>EXCLUSIVE-OR WITH TRANSLATE TABLE (02D6)</td>
</tr>
<tr>
<td><strong>C6A8</strong></td>
<td>DECREMENT OFFSET</td>
</tr>
<tr>
<td><strong>C6B5</strong></td>
<td>STORE BYTE IN AUXILIARY BUFFER (0300)</td>
</tr>
<tr>
<td><strong>C6B8</strong></td>
<td>LOOP UNTIL BUFFER FULL &gt;&gt;C6A</td>
</tr>
<tr>
<td><strong>C6B8</strong></td>
<td>INITIALIZE OFFSET (MAIN BUFFER)</td>
</tr>
<tr>
<td><strong>C6B8</strong></td>
<td>READ DATA REGISTER (C00C)</td>
</tr>
<tr>
<td><strong>C6B8</strong></td>
<td>LOOP UNTIL VALID &gt;&gt;C6CB</td>
</tr>
<tr>
<td><strong>C6D0</strong></td>
<td>IS CHECKSUM OKAY? (02D6)</td>
</tr>
<tr>
<td><strong>C6D3</strong></td>
<td>NO, START OVER &gt;&gt;C6A2</td>
</tr>
<tr>
<td><strong>C6D5</strong></td>
<td>MERGE MAIN AND AUXILIARY BUFFERS</td>
</tr>
<tr>
<td><strong>C6D9</strong></td>
<td>INITIALIZE OFFSET (MAIN BUFFER)</td>
</tr>
<tr>
<td><strong>C6DA</strong></td>
<td>INITIALIZE OFFSET (AUXILIARY BUFFER)</td>
</tr>
<tr>
<td><strong>C6DB</strong></td>
<td>DECREMENT OFFSET (AUX BUFFER)</td>
</tr>
<tr>
<td><strong>C6DB</strong></td>
<td>IF LESS THAN ZERO RESET IT &gt;&gt;C6D7</td>
</tr>
<tr>
<td><strong>C6DC</strong></td>
<td>GET BYTE FROM MAIN BUFFER</td>
</tr>
<tr>
<td><strong>C6DE</strong></td>
<td>ROLL IN TWO BITS FROM AUXILIARY BUFFER</td>
</tr>
<tr>
<td><strong>C6E6</strong></td>
<td>SAVE COMPLETED DATA BYTE</td>
</tr>
<tr>
<td><strong>C6E9</strong></td>
<td>LOOP UNTIL WHOLE BUFFER IS DONE &gt;&gt;C6D9</td>
</tr>
<tr>
<td><strong>C6EB</strong></td>
<td>DETERMINE IF THERE IS MORE TO DO</td>
</tr>
<tr>
<td><strong>C6EB</strong></td>
<td>INCREMENT MAIN BUFFER POINTERS</td>
</tr>
<tr>
<td><strong>C6ED</strong></td>
<td>INCREMENT SECTOR NUMBER</td>
</tr>
<tr>
<td><strong>C6F1</strong></td>
<td>IS THERE ANOTHER SECTOR TO LOAD? (8000)</td>
</tr>
<tr>
<td><strong>C6F6</strong></td>
<td>YES, GO DO IT &gt;&gt;C6D3</td>
</tr>
<tr>
<td><strong>C6FB</strong></td>
<td>NO, ENABLE INTERRUPTS &lt;&lt;C645</td>
</tr>
<tr>
<td><strong>C6FB</strong></td>
<td>AND JUMP TO CODE WE JUST LOADED. &gt;&gt;8001</td>
</tr>
</tbody>
</table>
Disk II Boot ROM -- Apple IIGS

NEXT OBJECT ADDR: 6409

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESCRIPTION/CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>6409</td>
<td>SET MSG CODE=0 (&quot;I/O ERROR&quot;)</td>
</tr>
<tr>
<td>640B</td>
<td>IS ERROR CODE ZERO? (E18FB1)</td>
</tr>
<tr>
<td>640F</td>
<td>NO, KEEP CHECKING &gt;&gt;6413</td>
</tr>
<tr>
<td>6411</td>
<td>YES, SET MSG CODE=1 (&quot;NOT A STARTUP DISK&quot;)</td>
</tr>
<tr>
<td>6413</td>
<td>IS IT &quot;NO DEVICE CONNECTED&quot;?</td>
</tr>
<tr>
<td>6415</td>
<td>NO. &gt;&gt;6419</td>
</tr>
<tr>
<td>6417</td>
<td>YES, SET MSG CODE=2</td>
</tr>
<tr>
<td>6419</td>
<td>IS IT &quot;CHECK STARTUP DEVICE&quot;?</td>
</tr>
<tr>
<td>641B</td>
<td>NO, CALL IT &quot;I/O ERROR&quot; &gt;&gt;641F</td>
</tr>
<tr>
<td>641D</td>
<td>YES, SET MSG CODE=3</td>
</tr>
<tr>
<td>641F</td>
<td>PUT MSG CODE IN ACCUM</td>
</tr>
<tr>
<td>6420</td>
<td>AND PRINT IT OUT &lt;C000&gt;</td>
</tr>
<tr>
<td>6423</td>
<td>INDICATE I/O ERROR</td>
</tr>
<tr>
<td>6425</td>
<td>GET P-REGISTER BACK</td>
</tr>
<tr>
<td>6426</td>
<td>RESTORE MODE BASED ON CARRY</td>
</tr>
<tr>
<td>6427</td>
<td>INDICATE ERROR</td>
</tr>
<tr>
<td>6428</td>
<td>RETURN TO $C600 LOGIC</td>
</tr>
</tbody>
</table>
APPENDIX A

THE PRODOS 8 VERSIONS

changes to ProDOS 8 that were...

This Appendix identifies the changes introduced with the 1.2 and 1.3 versions of ProDOS 8.

CHANGES INTRODUCED IN THE 1.2 VERSION

The changes that were made to produce ProDOS 8, Version 1.2, are:

1. When running on an IIGS with an external RAM disk, the relocators are... 2006, page 11.

2. The ProDOS 8 version number is included as the first two bytes of the directory header.

3. Always checks the aux_stack_prior_flag (2278). Also sets boot on a IIGS.

4. Sets aux_stack pointer

5. No 80-column card [212A-2139, page 12].

6. If operating on a IIGS, installs the IIGS Clock Code [21AD-21D0, 22D3-22DA, pages 12 and 14].

7. Now checks for an AppleTalk Initialization File (ATINIT file) before looking for a .SYSTEM file. If the ATINIT file is found, it is loaded and executed, then the search for a .SYSTEM file commences. [22DB-2381, page 14].

8. The list of device is now ordered differently. It recognizes the SmartPort and allows four Slot 5 SmartPort units to be accessed as Slot 5, Drives 1 and 2 and Slot 2, Drives 1 and 2. If Slot 2 is being used by a storage device, however, only the first two devices on the Slot 5 SmartPort can be accessed. It also changes the search order, making sure that Disk II devices are searched last when a device scan takes place (such as during an MLI ON LINE call). [2668-271B, 275D-2767, 2814-28AA, pages 17-19].
9. A bug in the /RAM driver (which we pointed out in the 1.1.1 supplement) that allowed a block read of block 7 (which doesn't exist) has been corrected [2D4B, page 23].

10. The /RAM caller, which operates in high RAM, now contains a $60 (RTS instruction) at address $FF58. Peripheral cards sometimes call that address to figure out which slot they are in, and in case they forget to set ROM for reading, the call will still work. [2E56-2E58, page 24].

11. A subroutine that sets high RAM for reading/writing was created to save space in the code [2518-251E, page 16].

MLI and MLI Global Page

1. The Global Page now pushes the P-Register and disables interrupts before calling the actual MLI [DE01-DE04, DE1C-DE21, BF4B-BF4F, pages 34 and 75].
2. Setting the MLIACTIVE flag a little differently now allows nested calls to the MLI by interrupt routines [DE8F-DE91, page 35].
3. A new MLI command was introduced (command=$82), that allows the user to install a routine to handle unclaimed interrupts [DEFF-DF0B, FD23-FD2C, pages 35 and 63].
4. If there is no unclaimed interrupt handler, ProDOS 8 now counts unclaimed interrupts, and will allow 255 of them to occur before finally issuing a fatal error. This allows a brief time for the unknown interrupt to stop interrupting. [DFB3-DFB7, page 36].
5. If operating on a IIGS and system death occurs, the NEWVIDEO softswitch is set to 0, reinitializing the IIGS video [E013-E016, page 36].
6. Processing for the ON_LINE command now frees the VCB entry for a device that was previously on-line but has been taken off-line [E28A-E2A4, page 39].
7. A subroutine that reads a block where the block number is in the A and X registers was added to save space in the code [EBC9-EBD0, page 47].
8. The error message that results when a file being opened has an illegal storage type has been changed from "incompatible directory format" to "unsupported storage type" [ECC7-EECA, page 50].
9. An error type $C is now indicated when truncating or deleting a file and the file's storage type is illegal [FA4D-FA4E, page 60].
10. To save space in the QuitCode Caller, some in-line code was changed to a loop [FCAF-FCB7, FCD8-FCE0, page 62].
Quit Code

1. Uses standard character set instead of alternate character set [1006-1008, page 77].
2. Sets normal 40-column screen in a safer way, such that screen hole values are preserved [100C-1011, page 77].
3. Message display routine is modified [1033-1034, 11D6-11D1, etc., pages 77 and 79].
4. The method of displaying the current prefix is changed so that it is always written to the same screen location [104D-105C, page 77].
5. User can now backspace with the DELETE key as well as left arrow [107C-107F, 10FC-10FF, page 78].
6. The method of inputting the Application name was modified [10E7-10E9, page 78].

Clock Code

1. The code for the ThunderClock includes a lookup table to determine the year based on the day of the year and the day of the week. This table is only good for a span of five or six years. The table released with Version 1.1.1 was good for the years 1982-1987. The table released with Version 1.2 covers the years 1986-1991. [D7B8-D7BE, page 91].
2. A completely separate clock routine is provided in Version 1.2 in case ProDOS 8 is operating on a IIGS. If so, the IIGS Clock Code is always enabled. It is written in 65816 and calls the ReadTimeHex tool in the tool kit to read the clock. [D742-D790, page 92].

CHANGES INTRODUCED IN THE 1.3 VERSION

The changes that were made to the 1.2 version of ProDOS 8 to produce ProDOS 8, Version 1.3, are listed below. Addresses given here are Version 1.3 addresses.

Relocator

1. The boot message now includes a line that says "ALL RIGHTS RESERVED." Chalk up one for the legal department! [25F6-2671, page 26].
2. A ProDOS Status call now immediately precedes the SmartPort Status Call. This is because the SmartPort interface does not set up its device list until it receives a ProDOS Status call. Earlier versions of ProDOS 8 may not always find all SmartPort devices. [286E-2894, page 28].
MLI

1. When files are deleted, previous versions of ProDOS zero out all but the first block of discarded index blocks. Now such index blocks will not be zeroed, but the pages of these blocks will be flipped. That is, the high byte of the block numbers will be exchanged with the low byte of the block numbers. [P992-P99A, FBC7-FBDC, pages 67 and 69].

2. Previous versions of ProDOS forgot, in certain cases, to rewrite index blocks that were being discarded when shortening or deleting files. Now such blocks will always be rewritten to disk in a zeroed (shortened file) or flipped (deleted file) form. [FAB8-FABC, FB91-FB93, pages 68 and 69].

3. A poorly-written loop in the QuitCode Caller that was added for Version 1.2 was rewritten for Version 1.3. The Version 1.2 code might cause problems on a IIGS. [FD05-FD0C, page 70].

Disk II Device Driver

1. There is a routine in the Disk II Device Driver that clears phases in case the Disk II device is sharing transmission lines with SmartPort devices. This routine was patched in Version 1.3 so that phases are now cleared with LDA instructions instead of STA instructions. This eliminates hangs that can, in some situations, cause unwanted writing to the floppy disk. [D6C3-D6CE, page 88].

BUGS IN VERSIONS 1.2 AND 1.3

It is fair to say that both Versions 1.2 and 1.3 of ProDOS 8 are relatively bug free. Perhaps a few escaped our notice, but we know of only three minor bugs, which are as follows:

MLI, Versions 1.2 and 1.3, at EC64 (see p. 47). This bug has been in ProDOS since day 1. Although there is no easy way to correct the problem (because a three-byte instruction is needed where there are only two bytes), any serious problems can be avoided by putting NOP's at EC64 and EC65 (3D64 and 3D65 in load location). This bug can only take effect when a storage type 0 is found (not likely unless disk swapping) and a lot of files are open simultaneously.

MLI, Version 1.2 only, at FCD8 (see p. 62). A loop that indexes around the 64K boundary may cause problems on a IIGS. Use version 1.3 or recode the loop so that the boundary crossover is eliminated.

MLI, Version 1.3 only, at FBCD (see p. 69). A 65C02 instruction snuck into the code, which will be disastrous when Version 1.3 is run on a computer with a 6502 processor (Apple II+, unenhanced IIe). It is easily patched, as we explain on page 69.
APPENDIX B

ERRATA TO BENEATH APPLE PRODOS

ERRATA TO BENEATH APPLE PRODOS (1st Printing, 1984)

You can identify which printing of Beneath Apple ProDOS you have by looking at the space between the title of the book and the author's names on the first page of the book (the title page). If this space is blank, you have the first printing. The second printing has "Second Printing, March 1985" in this space. If you have the second printing, skip to page 120. If you have the first printing, all of the following errata apply.

Page 3–16:

In the first paragraph starting on the page, the sentence should read "The data is dealt with in larger pieces (512 bytes vs. 256 bytes)...", not 512K vs. 256K.

Page 6–63:

The code for "HOW MUCH MEMORY IS IN THIS MACHINE?" is incorrect. Replace it with:

```
LDA $BF98           GET MACHID FROM GLOBAL PAGE
ASL A              MOVE BITS TO TEST POSITION
ASL A
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                PUT BIT PATTERN IN ACCUM
ORA BITMAP,Y       MARK THIS PAGE AS IN USE
STA BITMAP,Y       UPDATE MAP
...                WE'VE GOT IT NOW
```
LOCATE PHA
AND $07
TAY
LDX BITMASK,Y
PLA
LSR A
LSR A
TAY
Y-REG IS OFFSET INTO BITMAP
LDA BITPATTERN IN ACUM
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
DEVL LDA $BF32,X PICK UP LAST DEVICE NUM
AND #$70 ISOLATE SLOT
CMP #$30 SLOT = 3?
BEQ GOTS LT YES, CONTINUE
DEX
BPL DEVL CONTINUE SEARCH BACKWARDS
BMI NORAM CAN'T FIND IT IN DEVLST
GOTS LT LDA $BF32+1,X GET NEXT NUMBER
STA $BF32,X AND MOVE THEM FORWARD
INX
CPX $BF31 REACHED LAST ENTRY?
BNE GOTS LT NO, LOOP
DEC $BF31 REDUCE DEVCNT BY 1
LDA #0 ZERO LAST ENTRY IN TABLE
STA $BF32,X
CLC
BCC OKXIT BRANCH ALWAYS TAKEN
SKP 1
OLDVEC DW 0 OLD VECTOR SAVEAREA
To reinstall the /BIN driver, execute this subroutine:

* SEE IF SLOT 3 HAS A DRIVER ALREADY
  SKP 1
  HIMEM EQU $73          PTR TO BI'S GENERAL PURPOSE BUFFER
  SKP 1
  INSTALL LDX $BF31     GET DEVCNT
  INSLP LDA $BF32,X    GET A DEVNUM
    AND #$70        ISOLATE SLOT
    CMP #$30        SLOT 3?
    BEQ INSOUT    YES, SKIP IT
    DEX
    BPL INSLP      KEEP UP THE SEARCH
  SKP 1

* RESTORE THE DEVNUM TO THE LST
  SKP 1
  INSTALL LDX $BF31     GET DEVCNT AGAIN
  CPX #$0D          DEVICE TABLE FULL?
  BNE INSLP4       YOUR ERROR ROUTINE

ERROR ...

INSLP4 LDA $BF32-1,X MOVE ALL ENTRIES DOWN
  STA $BF32,X      TO MAKE ROOM AT FRONT
  DEX              FOR A NEW ENTRY
  BNE INSLP4
  LDA #$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:

<table>
<thead>
<tr>
<th>BF10-BF11</th>
<th>DEVADR01</th>
<th>Slot 0 reserved.</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>DEVADR32</td>
<td>/RAM device driver address</td>
</tr>
<tr>
<td>BF26-BF27</td>
<td></td>
<td>(need extra 64K).</td>
</tr>
</tbody>
</table>

Page 8-7:

The wrong bit is indicated as the "expansion bit" in the MACHID byte. The first eight rows of that description should read:

00.. 0... II
01.. 0... II+
10.. 0... IIE
11.. 0... III emulation
00.. 1... Future expansion
01.. 1... Future expansion
10.. 1... IIC
11.. 1... Future expansion

Page B-8:

In the last paragraph, the sentence should read "A second way to use an interpreted language..." (not a compiled language).

Page D-1:

In the second paragraph, the sentence should read "Versions of the Disk Drive Controller Unit are now used..." (not based).

Reference Card, Panel 4

Under "SYSTEM GLOBAL PAGE FORMAT", replace the lines beginning BF05 and BF06 with the following two lines:

BF06 Jump to Date/Time Address
(or RTS if no clock)
The description of BF10-11 should be changed to:

BF10-11 Slot 0 reserved

The description of BF26-27 should be changed to:

BF26-27 /RAM

Under the "MACHINE IDENTIFICATION BYTE", the second column of numbers should read:

0...
0...
0...
0...
1...
1...
1...
1...

Reference Card, Panel 9

The last entry for "MLI ERROR CODES" should be:

$5A Bad vol. bit map

(not $58).
ERRATA TO BENEATH APPLE PRODOS (2nd Printing, 1985)

Page 4–30:

The definitions of PARENT POINTER and PARENT ENTRY are incorrect. Replace them with:

$27–$28 PARENT_POINTER: The block number (within the volume directory or a subdirectory) which contains the file entry for this subdirectory.

$29 PARENT_ENTRY: The number of the file entry within the block number pointed to by the PARENT_POINTER. Given that "ENTRIES PER BLOCK" is $0D, then the PARENT_ENTRY number ranges from $01 to $0D.

Page 6–62:

The paragraph immediately preceding Table 6.6 should read as follows:

If an error occurs, the BI error code will be placed in the accumulator. Possible codes are listed in Table 6.6.

In Table 6.6, the message for error code $0C is wrong. It should read:

$0C NO BUFFERS AVAILABLE

Page 7–26:

Expand the 40-track drive patch to show how to patch all of the versions of ProDOS 8 released to date.

This patch modifies the Disk II Driver, which is a part of the "PRODOS" file (or "P8" file), so that it allows 320 blocks per volume instead of 280 blocks per volume. First set the prefix to the directory that contains the file you want to modify. This file will normally be called "PRODOS" on an 8-bit Apple II and "P8" on a 16-bit Apple II GS. If the file name is not "PRODOS," substitute the correct filename wherever "PRODOS" appears.

UNLOCK PRODOS
BLOAD PRODOS,TSYS,A$2000
CALL -151
address*:40
3D00
BSAVE PRODOS,TSYS,A$2000
LOCK PRODOS
ProDOS, as follows:

Program FILER to format 40-track drives may be formatted with 40-track drives 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>

In the listing of the "TYPE" program, change the value 4 to 5 as follows:

2115:C0 05 207 CPY #5
Quality Software Products For the Apple

BOOKS

Beneath Apple ProDOS by Don Worth & Pieter Lechner
Describes the ProDOS Operating System clearly and in detail, going beyond Apple's manuals. Many programming examples are included. 288 pages. $19.95

Supplements to Beneath Apple ProDOS:
Versions 1.0.1 and 1.0.2 (combined) $10.00
Version 1.1.1 $12.50
Versions 1.2 and 1.3 (combined) $12.50

Beneath Apple DOS by Don Worth & Pieter Lechner
The popular best seller that covers all facets of DOS 3.3 and previous Apple disk operating systems. 176 pages. $19.95

Understanding the Apple II by Jim Sather
Foreword by Steve Wozniak. A definitive source of information, covers Apple II and Apple II Plus hardware, including the disk controller and logic state sequencer. 352 pages. $22.95

Understanding the Apple IIe by Jim Sather
The companion to Understanding the Apple II, this book covers Apple IIe hardware, including video graphics and the 1982 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 Converter by Gary Charpentier
Moves programs and data among the five operating systems used on the Apple II family of computers: DOS, ProDOS, CP/M, Pascal, and SOS. Unprotected 5 1/4" diskette and 48-page manual. 64K. $34.95
Ordering directly from Quality Software

To order our products directly, mail this order form to Quality Software (at the address below) with your payment—the price of the software (plus sales tax if shipped to California) plus shipping and handling charges. Your payment can be a check or bank draft made payable to Quality Software in US dollars, or your VISA or MASTERCARD number and expiration date (VISA and MASTERCARD holders must include their complete 16-digit numbers). Sales tax must be included if the appropriate state sales tax (6%, 6.5%, or 7%).

Shipping charges:
- 48 Continental United States (UPS).................. $2.50
- Alaska, Hawaii, Canada, and Mexico (air mail)...... $5.00
- All other countries (insured air mail)................ $10.00

Send your order to:

**QUALITY SOFTWARE**
21610 Lassen Street #7
Chatsworth CA 91311
(818) 709-1721

<table>
<thead>
<tr>
<th>QUANTITY</th>
<th>DESCRIPTION</th>
<th>AMOUNT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SUBTOTAL</th>
</tr>
</thead>
</table>

(CA RESIDENTS) SALES TAX

SHIPPING

TOTAL

Check # ________

OR VISA/MasterCard # ___________________________ EXPIRES ________

Name ________________________________

Street Address ________________________________

City, State, Postal Code: __________

Country ________________________________

(1.2)
SUPPLEMENT TO

Beneath Apple ProDOS

For ProDOS 8, Versions 1.2 and 1.3

by Don Worth and Pieter Lechner

QS QUALITY SOFTWARE