

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

<b>Beneath Apple ProDOS</b> by Don Worth & Pieter Lechner	\$19.95
<b>Supplement to Beneath Apple ProDOS for Versions 1.0.1, 1.0.2</b> by Don Worth & Pieter Lechner	\$10.00
<b>Supplement to Beneath Apple ProDOS for Version 1.1.1</b> by Don Worth & Pieter Lechner	\$12.50
<b>Beneath Apple DOS</b> by Don Worth & Pieter Lechner	\$19.95
<b>Understanding the Apple II</b> by Jim Sather	\$22.95
<b>Understanding the Apple IIe</b> by Jim Sather	\$24.95

## Apple Utility Software from Quality Software

<b>Bag of Tricks 2 (includes diskette)</b> by Don Worth & Pieter Lechner	\$49.95
<b>Universal File Conversion (includes diskette)</b> by Gary Charpentier	\$34.95

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

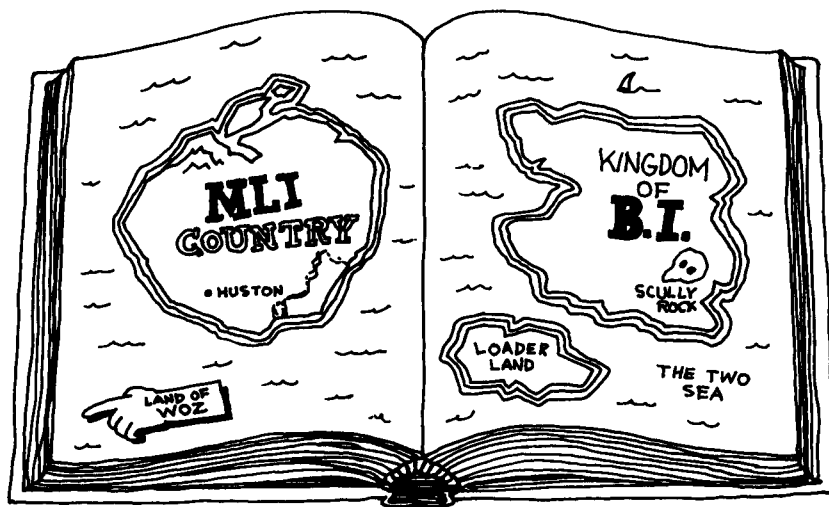
Illustrations by George Garcia

**(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 use of the term "Apple" should not be construed to represent any endorsement, official or otherwise, by Apple Computer, Inc.

## CONTENTS

<u>PAGE</u>	<u>TOPIC</u>
5	Introduction
5	Understanding the Listings
<b>PRODOS 8, VERSIONS 1.2 AND 1.3</b>	
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)
<b>BASIC.SYSTEM, VERSION 1.1</b>	
93	How BASIC.SYSTEM is Loaded and Relocated
94	BI Relocator
97	BASIC Interpreter (BI)
132	BI Global Page
<b>DISK II BOOT ROM</b>	
134	Disk II Controller ROM--Apple II/II+/IIe
136	Disk II Boot Logic--Apple IIc
139	Disk II Boot Logic--Apple IIGS
143	<b>APPENDIX A -- Differences Between ProDOS 8 Versions</b>
147	<b>APPENDIX B -- Errata to Beneath Apple ProDOS</b>



# A ProDOS ATLAS

## INTRODUCTION

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

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

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

## UNDERSTANDING THE LISTINGS

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

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

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

Page titles give the address of the next instruction or data area in the module to be described. These may be used to quickly locate a particular area within the component.

HOW PRODOS 8 Versions 1.2 and 1.3 ARE LOADED AND RELOCATED

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

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

I-----I	I\$FFFF
I  IRQ HANDLER  I	I
I-----I	I\$FF9B
I  /RAM CALLER  I	I
I-----I	I\$FF00
I                  I	I
I          MLI          I	I
I                  I	I
I          KERNEL          I	I
I                  I	I
I  (run location)  I	I
I                  I	I
I-----I	I\$DE00
I  MLI DATA AREA  I	I
I-----I	I\$D700
I  DISK II DRIVER  I	I
I-----I	I\$D000
I                  I	I
I-----I	I\$C000
I  SYSTEM GLOBAL PAGEI	I
I-----I	I\$BF00
:                  :	:
:                  :	:
I-----I	I\$5C7E
I  IIGS CLOCK CODE  I	I
I-----I	I\$5C00
I  QUIT CODE          I	I
I-----I	I\$5900
I  DISK II DRIVER  I	I
I-----I	I\$5200
I  IRQ HANDLER      I	I
I-----I	I\$519B
I  CLOCK CODE          I	I
I-----I	I\$5100
I  SYSTEM GLOBAL PAGEI	I
I-----I	I\$5000
I                  I	I
I          MLI          I	I
I                  I	I
I  (load location)  I	I
I                  I	I
I-----I	I\$2F00
I  RELOCATOR          I	I
I-----I	I\$2000
:                  :	:
:                  :	:
:                  :	:
I-----I	I\$96C
I  SYSTEM FILE LOADER  I	I
I-----I	I\$800
I  PAGE 3 IMAGE      I	I
I-----I	I\$400
I  PAGE 3 IMAGE      I	I
I-----I	I\$3D6
I                  I	I
I-----I	I\$C8
I  80-COL CARD CHECKER  I	I
I-----I	I\$80
I  Caller          2E00  FF00  9A	I
I  Driver          2C00  200***200	I
I-----I	I\$0

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

- ④ Final moves:

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

\*5C00 if IIGS    \*\*BANK2    \*\*\*AUX MEMORY

ProDOS Loader -- V1.2 -- 6 SEP 86 NEXT OBJECT ADDR: 0800  
 ADDR DESCRIPTION/CONTENTS

ProDOS Loader -- V1.2 -- 6 SEP 86 NEXT OBJECT ADDR: 0800  
 ADDR DESCRIPTION/CONTENTS

```

0800 MODULE STARTING ADDRESS
*****
* PRODOS LOADER
* THIS CODE IS LOADED FROM BLOCK 0
* INTO MEMORY AT $800.
* ITS PURPOSE IS TO LOAD THE "PRODOS"
* FILE INTO $2000 AND JUMP TO IT.
* (PRODOS RELOCATOR IS AT $2000)
*
* VERSION 1.2 -- 6 SEP 86
* VERSION 1.3 -- 2 DEC 86
* (THE LOADER IS STILL THE SAME AS IT
* WAS IN VERSION 1.0.1)
*
*****
  
```

```

C08C SHIFT DATA REGISTER
FC58 HOME CURSOR/CLEAR SCREEN
*****
0800 SIGNATURE BYTE ($01 MEANS BOOT ROUTINE FOLLOWS)
(A $03 IS STORED HERE DURING A 5.25" FLOPPY BOOT)
-- APPLE /// BOOTING --
THIS CODE (BLOCK 0) IS LOADED AT $A000 WHEN
BOOTED ON AN APPLE ///. THE APPLE /// BOOT
ROM JUMPS TO $A000. WHAT IS SHOWN HERE AS
$800 ON AN APPLE II IS $A000 ON AN APPLE ///.
THUS AN APPLE /// EXECUTES A HARMLESS
INSTRUCTION (ORA $B,X), THEN DOES NOT BRANCH
ON CARRY, AND JUMPS TO $A132 ($932 ON AN
APPLE II). MANY THANKS TO DAV HOLLE FOR
PROVIDING US WITH THIS APPLE /// INFORMATION.
  
```

```

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

```

0801 ***** MAIN ENTRY *****
ON ENTRY, X = SLOT*16
A = SECTOR NUMBER
0801 ENTRY POINT FOR APPLE II
0802 ALWAYS TAKEN (APPLE II) >>0807
0804 JUMP TO APPLE /// LOGIC >>A132
0807 SAVE SLOT*16
0809 READING SECTOR 3 NEXT?
080B REMEMBER THIS...
080D MAKE $CX FROM SLOT*16
0815 AND SAVE AT $49
0819 $48/49 --> $CXFF IN ROM BOOT
081C CHECK $CXFF
081D BOOT ROM FOR DISK II?
081F NO, NOT A 5.25" FLOPPY >>085B
0821 GOT BOTH SECTORS OF LOADER? >>0831
0823 NO, STOP AT SECTOR 3
0825 STORE ON PARM (0800)
0828 SKIP SECTOR 1 (GET SEC 2)
082A DUMMY UP $CX5C AS RETURN ADDRESS
0830 AND CALL ROM SECTOR READ SUBRTN
  
```

```

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

```

***** LOAD PRODOS *****
(ENTIRE LOADER IN MEMORY NOW)
0831 CURRENT TRACK IS ZERO
0833 $4B/49 --> $CX00
0837 COPY A PORTION OF DISKETTE BOOT ROM
0839 TO MY BLOCK READER SUBROUTINE (0994)
083D FROM $9F7 TO $A7E
0843 MODIFY SOME BRANCHES IN THE COPIED CODE (091D)
0846 TO SUIT MY ERROR HANDLING TASTES (0924)
  
```

ProDOS Loader -- V1.2 -- 6 SEP 86      NEXT OBJECT ADDR: 084C  
 ADDR    DESCRIPTION/CONTENTS

```

084C AND COPY SECTOR READ SUBROUTINE EXIT CODE (092B)
084F TO $A7F TO $A85 (0A7F)
0855 $48/49 --> DISKETTE BLOCK READER SUBRTN
0859 AT $0986
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 --> $C00 (BUFFER)
0877 $4A/4B --> $C00 (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 6?
0888 NO, GO READ NEXT ONE >>0879
088A YES, CHECK LINK FOR VALIDITY (0C00)
088D IT SHOULD BE ZERO FOR VOL DIR (0C01)
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
0898 ---
0899 ADD ENTRY LENGTH TO FIND NEXT ENTRY (0C23)
089D STILL IN SAME PAGE? >>08AC
089F NO, BUMP ENTRY MSB
08A3 IS IT ODD? (SECOND PAGE OF A BLOCK?)
08A4 YES... >>08AC
08A6 NO, JUST FINISHED LAST BLOCK?
08A8 YES, ERROR -- FILE NOT FOUND >>08FF
08AA ELSE, START JUST PAST LINKS
08AC UPDATE LSB OF ENTRY POINTER
08AE GET NAME LENGTH (0902)
08B1 MASK OFF STORAGE TYPE
08B4 COMPARE NAME WITH "PRODOS"
08B9 NOT A MATCH? >>0896
08BE IF NAME MATCHES, IS IT A SAPLING FILE?
08C2 IF NOT, I CAN'T HANDLE IT >>08FF
08C6 GET FILE TYPE
08C8 SHOULD BE A PRODOS SYS FILE
08CA IF NOT, I GIVE UP >>08FF
08CD ALL IS WELL, COPY KEY BLOCK NUMBER
08CF TO $46/47
08D6 $4A/4B AND $60/61 --> $1E00
08D8 (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  
 ADDR    DESCRIPTION/CONTENTS

```

08EA BUMP TO NEXT BLOCK BUFFER
08EE $4E = OFFSET INTO INDEX BLOCK
08F0 GET NEXT BLOCK NUMBER FROM INDEX BLOCK
08F8 BLOCK NUMBER = 0? (END OF FILE)
08FA NOT YET, READ A BLOCK >>08E3
08FC ELSE, JUMP TO RELOCATOR AT $2000 >>2000
08FF ERROR JUMP >>093F
0902 ***** KERNEL NAME *****
0902 LENGTH OF KERNEL'S NAME
0903 'PRODOS', "PRODOS" (KERNEL NAME)
0912 ***** COPY BLOCK READ BUFFER PTR *****
0912 COPY $60/61 --> $44/45
0914 (BLOCK READ BUFFER POINTER)
091A THEN GO TO BLOCK I/O ROUTINE >>0048
091D ***** ROM SECTOR READ OFFSETS *****
      OFFSETS INTO ROM SECTOR READ SUBROUTINE
      TO BRANCH DISPLACEMENTS WHICH NEED TO
      BE CHANGED FOR LOADER'S PURPOSES
091D *****
0924 ---
      ***** NEW BRANCH OFFSETS FOR ABOVE ***
092B ***** SECTOR READ EXIT CODE *****
      COPIED TO END OF DISKETTE SECTOR READ CODE
092B GET SLOT*16
092E AND EXIT NORMALLY
092F RETURN
092F RESTART BLOCK READ OPERATION >>09BC
0932 ***** APPLE /// BOOT CODE *****
A132 THIS IS $A132 WHEN BOOTED ON APPLE ///
0932 MAKE IT LOOK LIKE A JSR FROM $A000
0938 LOAD IN BLOCK 1 (WE WANT SOS, NOT PRODOS)
093C GO TO APPLE /// BLOCK READ ROUTINE >>F479
  
```



ProDOS Loader -- V1.2 -- 6 SEP 86 NEXT OBJECT ADDR: 093F

ProDOS Loader -- V1.2 -- 6 SEP 86 NEXT OBJECT ADDR: 09D0

ADDR DESCRIPTION/CONTENTS

ADDR DESCRIPTION/CONTENTS

```

093F ***** ERROR HANDLER *****
093F HOME CURSOR/CLEAR SCREEN <FC58>
0944 COPY "UNABLE TO LOAD PRODOS" MESSAGE (0950)
0947 TO SCREEN (09AE)
094D THEN GO TO SLEEP FOREVER >>094D
0950 ---
0950 ***** UNABLE TO LOAD PRODOS *****
096D ***** MOVE ARM TO NEXT PHASE *****
096D GET CURRENT PHASE
096F CONVERT TO NEXT ARM PHASE
0972 ADD SLOT*16
0975 SELECT NEXT ARM PHASE THIS DRIVE (C080)
097A ---
097C DELAY LONG ENOUGH FOR ARM TO MOVE
0983 WHEN FINISHED, RETURN WITH X = SLOT*16
0985 RETURN
    
```

```

09D0 NO, ADJUST PHASE UP...
09D4 OR DOWN AND...
09D6 ---
09D7 SEEK ARM ONE PHASE... <096D>
09DD IN PROPER DIRECTION <096F>
09E0 UNTIL WE ARE THERE >>09C5
09E2 ---
09E4 RETRY COUNT OF 127
09E7 ---
09E9 LOWER RETRY COUNT
09EB RETRIES EXHAUSTED? >>09BB
09EF RETRIES FOR A $D5 HEADER
09F2 CHECK DATA REGISTER (C08C)
09F5 LOOP UNTIL DATA IS VALID >>09F2
    
```

```

***** SECTOR READ ROUTINES *****
09F7 BEGINNING OF COPIED ROUTINES
(SEE $C65E IN BOOT FIRMWARE DESCRIPTIONS)
($CX63-$CXEA IS COPIED TO $9F7-$A7E)
09A7F EXIT CODE FOR READ ROUTINES
(COPIED HERE FROM $92B-$930)
09A86 ***** $A86-$BFF NOT USED *****
09A86 NOT USED
09C00 ***** VOLUME DIRECTORY BUFFER *****
09C00 START OF VOLUME DIRECTORY BUFFER
09C23 OFFSET TO ENTRY LENGTH FIELD
    
```

```

***** DISKETTE BLOCK READ ROUTINE *****
$44/$45 --> BUFFER
$46/$47 = BLOCK NO.
0986 GET BLOCK NO. LSB
0988 ISOLATE SECTOR REMAINDER
098C SKEW SECTOR BY 2
0992 AND STORE SECTOR WANTED
0994 GET MSB
0996 AND HIGH BIT OF TRACK
0999 MERGE WITH LOW PART OF TRACK
099C STORE TRACK WANTED
099F TRACK*2 IS PHASE WANTED
09A3 SET PAGE ADDRESS OF BUFFER
09A7 TURN DRIVE MOTOR ON (C089)
09AA READ SECTOR <09BC>
09AD NEXT PAGE
09B1 SKEW TO NEXT SECTOR
09B5 READ SECOND SECTOR OF BLOCK <09BC>
09B8 THEN TURN MOTOR OFF AND EXIT (C088)
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
    
```

ProDOS Relocator -- V1.2 -- 6 SEP 86      NEXT OBJECT ADDR: 2000  
 ADDR    DESCRIPTION/CONTENTS

2000    MODULE STARTING ADDRESS  
 \* \* \* \* \*  
 \*    PRODOS RELOCATOR \* \* \* \* \*  
 \*    LOADED AS THE FIRST \* \* \* \* \*  
 \*    PORTION OF THE PRODOS \* \* \* \* \*  
 \*    IMAGE AT \$2000. \* \* \* \* \*  
 \*    VERSION 1.2 -- 6 SEP 86 \* \* \* \* \*  
 \* \* \* \* \*

\*\*\*\*\* ZERO PAGE ADDRESSES \*\*\*\*\*  
 000A    AUTOSTART ROM CHECKSUM POINTER  
 000B    CONFIGURATION BYTE (MACHID TO BE)  
 000C    GENERAL PURPOSE POINTER  
 0010    DISK TYPE (0=DISK II, 4=PROFILE)  
 0011    AND INPUT RELOC RANGE POINTER  
 0012    VOL DIR ENTRY POINTER FOR RELOCATOR  
 0013    AND OUTPUT RANGE PTR  
 0014    LENGTH OF RELOCATION RANGE  
 0015    INPUT RELOCATION RANGE POINTER  
 0016    END OF INPUT RANGE  
 0017    GENERAL PURPOSE POINTER  
 0018    GENERAL PURPOSE POINTER  
 0019    GENERAL PURPOSE POINTER  
 001A    GENERAL PURPOSE POINTER  
 003C    RAMDRIVE OUTPUT POINTER  
 003D    VARIOUS USES: PARM TO AUXMOVE,  
 003E    UNIT/SLOT PASSED TO RELOCATOR  
 003F    BLOCK NUMBER TO RAMDRIVE  
 0040    \*\*\*\*\* EXTERNAL ADDRESSES \*\*\*\*\*  
 0041    MACHID BUILD SUBRTN FOR 128K  
 0042    SAVE AUX STACK POINTER (IN AUX STACK)  
 0043    GENERAL PURPOSE BUFFER  
 0046    BUFFER+1  
 0047

ProDOS Relocator -- V1.2 -- 6 SEP 86      NEXT OBJECT ADDR: 2000  
 ADDR    DESCRIPTION/CONTENTS

\*\*\*\*\* SCREEN LINE ADDRESSES \*\*\*\*\*  
 04B8    SCREEN BUFFER ROW 10  
 05A9    SCREEN BUFFER ROW 12  
 05AD    SCREEN BUFFER ROW 14  
 06B6    SCREEN BUFFER ROW 16  
 07A8    SCREEN BUFFER ROW 18  
 07AD    SCREEN BUFFER ROW 20  
 07D0    SCREEN BUFFER ROW 24

\*\*\*\*\* MISCELLANEOUS ADDRESSES \*\*\*\*\*  
 0800    ENTRY OF INTERP LOADER  
 0C00    VOLUME DIRECTORY BUFFER  
 0C23    ENTRY LENGTH  
 -- RAMDRIVE VOLUME DIRECTORY --  
 0E04    VOLUME HDR, VOLUME NAME  
 0E22    VOLUME HDR, ACCESS-TOTAL BLOCKS  
 2000    START OF SYSTEM PROGRAMS  
 2C00    RAMDRIVE DEVICE DRIVER LOAD ADDRESS  
 2A00    DIFFERENCE OF RAMDRIVE LOAD AND RUN LOCATIONS  
 BFFF    TOP OF 48K 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 CONTAIN 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

ProDOS Relocator -- V1.2 -- 6 SEP 86 NEXT OBJECT ADDR: 2000

ADDR DESCRIPTION/CONTENTS

C030 SPEAKER  
 C054 USE MAIN MEMORY PART OF 80-COL CARD  
 C055 USE AUX MEMORY PART OF 80-COL CARD  
 C068 IIGS STATEREG STATUS BYTE  
 C081 WRITE-ENABLE HIGH RAM  
 C082 MOTHERBOARD ROM READ ENABLE  
 C083 READ/WRITE RAM 2ND 4K BANK  
 C08B READ/WRITE RAM 1ST 4K BANK

\*\*\*\*\* INTERNAL C3ROM ADDRESSES \*\*\*\*\*  
 C311 MOVE TO/FROM AUXMEM SUBROUTINE  
 C314 TRANSFER TO/FROM AUXMEM SUBROUTINE

\*\*\*\*\* SLOT ROM ADDRESSES \*\*\*\*\*  
 C305 SLOT3 I.D. BYTE  
 C307 SLOT3 I.D. BYTE  
 C30B SLOT3 I.D. BYTE  
 C30C SLOT3 I.D. BYTE  
 C3FA SLOT3 I.D. BYTE  
 CFFF RESET I/O CARD ROMS

\*\*\*\*\* PRODOS ADDRESSES \*\*\*\*\*  
 D000 START OF QUITCODE MEMORY AREA (BANK2)  
 DF11 ENHANCED ROM FLAG  
 FDB8 VERSION NUMBER (FOR SUBDIRECTORIES)  
 FEFF GS VIDEO FLAG  
 FF00 RAMDRIVE CALLER ADDRESS

\*\*\*\*\* MONITOR ROM \*\*\*\*\*  
 FB1E PADDLE READ SUBROUTINE  
 FB2F MONITOR INIT ROUTINE  
 FBB3 ROM VERSION BYTE  
 FBC0 SECONDARY VERSION BYTE (0-3)  
 FC58 CLEAR SCREEN  
 FELF THIS ROUTINE CHECKS FOR IIGS  
 FE84 SET NORMAL VIDEO  
 FE89 IN#0  
 FE93 PR#0

\*\*\*\*\* PRODOS RELOCATOR MAIN ENTRY \*\*\*\*\*  
 2000 JUMP OVER PQUIT ENTRY >>2006  
 2003 SET FLAG INDICATING PQUIT ENTRY (IIGS) (21D1)  
 2006 STORE SLOT IN MLI ONLINE PARMS  
 200B PRINT "APPLE II PRODOS..." <25B1>  
 200E SET UP FOR COMMON MOVES (226E)  
 2014 RELOCATE SOME ROUTINES & DATA TO LOW MEMORY <28B0>

ProDOS Relocator -- V1.2 -- 6 SEP 86 NEXT OBJECT ADDR: 2017

ADDR DESCRIPTION/CONTENTS

2017 RELOCATION ERROR >>203C  
 201D BE SURE 48K OF MAIN MEMORY EXISTS (BFFF)  
 2024 IF NOT, ERROR >>204E  
 2029 MAKE DOUBLY SURE (BFFF)  
 202C ERROR THIS TIME >>204E  
 202E SELECT MOTHERBOARD ROMS (C082)  
 2031 DETERMINE MACHINE TYPE <251F>  
 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 <28B0>  
 2048 GET PRODOS VERSION NUMBER (BFFF)  
 204B AND PUT IT IN MLI DATA AREA. (FDB8)  
 204E RELOCATION ERROR! >>204C  
 2050 ENABLE MOTHERBOARD ROMS AGAIN (C082)  
 2053 CHECK ROM I.D. BYTE (FBB3)  
 2056 APPLE //e FAMILY?  
 2058 NO, LEAVE I.D. BYTE AS IS >>208E  
 205C TEST ANOTHER ROM I.D. BYTE (FBC0)  
 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 \$FBC0 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? <FELF>  
 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  
 NOW SET OS BOOT TO ZERO, INDICATING THAT PRODOS8  
 WAS THE OPERATING SYSTEM INITIALLY BOOTED.

208A 65816 INSTRUCTION: STA \$E100BD  
 208E COPY BOOT DEVICE ID TO READ BLOCK PARMS (2262)  
 2094 AND AS LAST DEVICE USED (BF30)  
 2097 DETERMINE PERIPHERAL CARD CONFIGURATION <265F>  
 209A BOOT DEVICE TO... (2269)  
 209D GLOBAL PAGE LAST DEVICE USED (BF30)  
 20A0 ENABLE READ/WRITE HIGH RAM, BANK 1 <2518>  
 20A9 COPY CLOCK CODE TO DEVICE DRIVER AREA <28B0>

ProDOS Relocator -- V1.2 -- 6 SEP 86      NEXT OBJECT ADDR: 20AC  
 ADDR    DESCRIPTION/CONTENTS

20AC ERROR? >>20DA  
 20AE CHECK MACHINE TYPE AGAIN (BF98)  
 20B1 GOT 64K OR MORE?  
 20B5 NO >>20DD  
 20B7 YES, QUIT VECTOR ---> \$FCA9  
 20C1 WRITE TO HIGH RAM (BANK2) (C083)  
 20CA POINT TO QUIT CODE TABLE (2275)  
 20CD MOVE QUIT CODE TO HIGH RAM <28B0>  
 20D2 STORE QUIT VECTOR START PAGE (D000)  
 20D5 ENABLE READ/WRITE HIGH RAM, BANK 1 <2518>  
 20DA RELOCATION ERROR >>2227  
 20DD GET MACHID YET AGAIN (BF98)  
 20E0 128K?  
 20E4 NO... >>20FC  
 20E6 YES, SET UP AUX RAM  
 20E8 SAVE STATUS REG IN ACCUM  
 20EA DISABLE INTERRUPTS  
 20EB PREPARE TO WRITE TO AUX STACK AREA (C009)  
 20EE AUX STACK POINTER SET TO \$FF (0101)  
 20F1 BACK TO MAIN Z-PAGE, STACK (C008)  
 20F4 RESTORE STATUS REG  
 20F9 ESTABLISH RAM DRIVE IN AUX MEM <2B00>  
 \*\*\*\*\* SET UP FOR IRQ (ENHANCED ROM) \*\*  
 20FC READ ROM (C081)  
 20FF GET ROM'S IRQ VECTOR (FFFE)  
 2105 ENABLE READ/WRITE HIGH RAM, BANK 1 <2518>  
 2108 CARRY CLEAR IF IRQ VECTOR IN C3 ROM  
 210A FLAG FOR "OLD ROM"  
 210C IT'S AN OLD ROM >>2127  
 210E SWITCH TO AUX STACK & HIGH RAM (C009)  
 2113 INITIALIZE AUX STACK POINTER TO \$FF (0101)  
 2116 PUT IRQ VECTOR IN AUX HIGH RAM (FFFF)  
 211C BACK TO MAIN HIGH RAM & Z-PAGE (C008)  
 211F PUT IRQ VECTOR IN MAIN HIGH RAM (FFFF)  
 2125 INDICATE ENHANCED IRQ LOGIC ON BOARD  
 2127 STORE FLAG IN MLI DATA AREA (DFF1)  
 \*\*\*\*\* LOOK FOR SLOT 3 VIDEO CARD \*\*\*\*\*  
 ---  
 212A SET GS VIDEO FLAG=0 (FFFE)  
 212C THIS A IIGS? (2278)  
 2132 NO. >>213A  
 2134 YES, SET GS VIDEO FLAG (IN MLI) (FFFE)  
 2137 AND DON'T BOTHER SEARCHING SLOT 3. >>21A5  
 213A ENABLE INTERNAL VIDEO FIRMWARE (C00A)  
 213D CHECK FOR ROM (BF99)  
 2140 IN SLOT 3.  
 2142 ROM EXISTS. >>2147

ProDOS Relocator -- V1.2 -- 6 SEP 86      NEXT OBJECT ADDR: 2144  
 ADDR    DESCRIPTION/CONTENTS

2144 NO ROM. >>21AD  
 2147 LOOK AT THE SLOT 3 ROM (C00B)  
 214A AT OFFSET +\$05 (C305)  
 214D THERE MUST BE A \$38  
 2151 AND AT OFFSET +\$07 (C307)  
 2154 THERE MUST BE AN \$18  
 2158 AND AT OFFSET +\$0B (C30B)  
 215B THERE MUST BE A 1  
 215F AND AT OFFSET +\$0C (C30C)  
 2164 INDICATE AN 80-COL CARD.  
 2168 CHECK MACHINE TYPE (BF98)  
 216D IS THIS AN APPLE III?  
 216F OK, IT'S GOT 80-COL CAPABILITY >>21A5  
 2171 OTHER MANUFACTURERS MUST FOLLOW THE RULES! (C3FA)  
 2174 MUST HAVE BIT INSTRUCTION AT \$C3FA  
 2176 GOOD BOY, YOU FOLLOWED THE RULES! >>21A5  
 2178 GIVE CONTROL BACK TO MOTHERBOARD ROM (C00A)  
 217B TURN ON 80-COL (C001)  
 217E CHECK FOR AUX MEM. (C055)  
 2183 PUT A BYTE AT AUX \$400 (0400)  
 2186 THE ACCUMULATOR LEFT  
 2187 AND DO THE SAME WITH \$400 (0400)  
 218A STILL THE SAME? (0400)  
 218D NO, NO 80-COL MEMORY >>2196  
 218F SHIFT TO THE RIGHT  
 2193 STILL THE SAME? (0400)  
 2196 BACK TO MAIN MEMORY (C054)  
 2199 TURN OFF 80-COL (C000)  
 219C WAS 80-COL MEMORY FOUND? >>21A5  
 219E NO, SO TURN OFF 80-COL FLAG (BF98)  
 21A1 IN MACHINE I.D. BYTE.  
 21A3 ALWAYS BRANCH >>21AA  
 21A5 TURN ON 80-COL FLAG (BF98)  
 21AD THIS A IIGS? (2278)  
 21B0 NO. >>21C8  
 21B2 YES, ENABLE IIGS CLOCK DRIVER  
 21B7 GET ADDRESS OF RELOCATE TABLE (2276)  
 21BA FOR IIGS CLOCK CODE (2277)  
 21BD AND PUT THE CODE AT \$D742 <28B0>  
 21C0 INDICATE CLOCK EXISTS IN MACHID (BF98)  
 21C8 ENTER FROM PQUIT? (21D1)  
 21CB NO. >>21D2  
 21CD YES, ENABLE ROM FOR READ (C082)  
 21D0 RETURN  
 21D1 PQUIT FLAG. (0 = PRODOS 8 WAS INITIAL BOOT)

ProDOS Relocator -- V1.2 -- 6 SEP 86      NEXT OBJECT ADDR: 21D1  
 ADDR    DESCRIPTION/CONTENTS

```

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

```

***** READ VOLUME DIRECTORY *****
--- $14/15 --> $C00
21FB 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
221E ADD TWO PAGES (ONE BLOCK) TO POINTER
2220 AND STOP AT $1400 IN ANY CASE
2222 ELSE, READ NEXT BLOCK AS WELL >>21FB
2224 WHEN DONE, JUMP TO SYSTEM FILE LOADER >>0900
  
```

```

2227 ***** ERROR HANDLER *****
2227 ENABLE MOTHERBOARD ROMS (C082)
222A CLEAR SCREEN <FC58>
222F PRINT "RELOCATION/CONFIG ERROR" (223B)
2238 THEN SLEEP FOREVER >>2238
  
```

```

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

ProDOS Relocator -- V1.2 -- 6 SEP 86      NEXT OBJECT ADDR: 226C  
 ADDR    DESCRIPTION/CONTENTS

```

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

```

2279 ***** RELOCATION TABLES *****
+0:
00 - ZERO BLOCK OF MEMORY
01 - COPY BLOCK
02 - RELOCATE MSB ADDRESSES
03 - RELOCATE 2 BYTE ADDRKS
04 - RELOCATE INSTRUCTIONS
+1/2: ADDR OF OUTPUT BLOCK
+3/4: LENGTH OF BLOCK IN BYTES
+5/6: ADDR OF INPUT BLOCK (IF ANY)
+7:    NUM RANGES TO CORRECT FOR (-1)
+8:    START PAGES
+8+COUNT:    END PAGE ADDRESSES
+8+COUNT+COUNT: ADDITIVE CORRECTION FACTOR
***** COMMON MOVES TABLE *****
  
```

```

2279 COPY (SYSTEM FILE LOADER)
227A    TO =$800
227C    LEN=$213
227E    FRM=$22DB
2280 COPY (PAGE 3 IMAGE)
2281    TO =$3D6
2283    LEN=$2A
2285    FRM=$24EE
2287 COPY (CHECKSUM)
2288    TO =$0A
228A    LEN=$02
228C    FRM=$14
228E COPY (CHECK FOR 80-COL CARD)
228F    TO =$80
2291    LEN=$46
2293    FRM=$256B
2295 END OF TABLE
***** QUIT CODE MOVE TABLE *****
2296 COPY (QUIT CODE)
2297    TO =$D100
2299    LEN=$300
229B    FRM=$5900
229D END OF TABLE
  
```

ProDOS Relocator -- V1.2 -- 6 SEP 86      NEXT OBJECT ADDR: 229D  
 ADDR    DESCRIPTION/CONTENTS

\*\*\*\*\* PRODOS RELOC TABLE \*\*\*\*\*

229E COPY (IRQ HANDLER)  
 229F    TO =\$FF9B  
 22A1    LEN=\$65  
 22A3    FRM=\$519B  
 22A5 COPY (SYSTEM GLOBAL PAGE)  
 22A6    TO =\$BF00  
 22A8    LEN=\$100  
 22AA    FRM=\$5000  
 22AC ZERO (PRODOS KERNEL DATA AREA)  
 22AD    ADR=\$D700  
 22AF    LEN=\$700  
 22B1 COPY (PRODOS KERNEL)  
 22B2    TO =\$DE00  
 22B4    LEN=\$2100  
 22B6    FRM=\$2F00  
 22B8 COPY (DISKETTE DRIVER)  
 22B9    TO =\$D000  
 22BB    LEN=\$700  
 22BD    FRM=\$5200  
 22BF END OF TABLE

\*\*\*\*\* THUNDERCLOCK TABLE \*\*\*\*\*

22C0 COPY (THUNDERCLOCK CODE)  
 22C1    TO =\$D742  
 22C3    LEN=\$7D  
 22C5    FRM=\$5100  
 22C7 RELOCATE INSTRUCTIONS  
 22C8    TO =\$D742  
 22CA    LEN=\$69  
 22CC    FRM=\$D742  
 22CE    FOR ADRES=\$C1XX-\$C1XX  
 22D1    ADJUST BY=\$S0  
 22D2 END OF TABLE

\*\*\*\*\* IIGS CLOCK TABLE \*\*\*\*\*

22D3 COPY (IIGS CLOCK CODE)  
 22D4    TO =\$D742  
 22D6    LEN=\$7D  
 22D8    FRM=\$5C00  
 22DA END OF TABLE

ProDOS Relocator -- V1.2 -- 6 SEP 86      NEXT OBJECT ADDR: 22DB  
 ADDR    DESCRIPTION/CONTENTS

22DB \*\*\*\*\* SYSTEM FILE LOADER \*\*\*\*\*  
 (COPIED TO AND RUN AT \$800)

22DB MLI: GET.FILE.INFO CALL <BF00>  
 22DE FIRST SEE IF THERE IS AN A "ATINT" FILE  
 22E1 NO ERRORS >>22EA  
 22E3 IS ERROR "FILE NOT FOUND"?  
 22E5 YES, THAT'S OK. >>232C  
 22E7 NO, OTHER ERROR >>232F  
 22EA GET FILE TYPE OF FILE FOUND (088D)  
 22ED IS IT "ATINIT" FILE?  
 22EF NO, ERROR >>232F  
 22F1 MLI: OPEN CALL <BF00>  
 22F7 FILE DOESN'T OPEN >>232F  
 22F9 MLI: GET.EOF CALL <BF00>  
 22FF CAN'T FIND EOF >>232F  
 2301 HIGH BYTE OF EOF (0A01)  
 2304 FILE TOO BIG >>232F  
 2306 MEDIUM BYTE OF EOF (0A00)  
 2309 MAX FILE SIZE IS \$9800  
 230B FILE TOO BIG >>232F  
 230D PUT IN READ PARM (0A07)  
 2310 GET LOW BYTE OF EOF (09FF)  
 2313 PUT IN READ PARM (0A06)  
 2316 MLI: READ CALL <BF00>  
 231C READ ERROR >>232F  
 231E MLI: CLOSE CALL <BF00>  
 2324 CLOSE ERROR >>232F  
 2326 READ ROM (C082)  
 2329 GO TO APPLICATION <2000>  
 232C NOW LOOK FOR SYSTEM FILE >>08A8  
 232F PRINT ERROR MESSAGE: (233D)  
 2332 "UNABLE TO LOAD ATINIT FILE" (233D)  
 233B SLEEP FOREVER >>233B  
 233D MSG LENGTH  
 233E "\*\*\* UNABLE TO LOAD ATINIT FILE \*\*\*"  
 2364 GET FILE INFO PARM (FOR ATINIT FILE)  
 (LOCATED AT \$889 WHEN EXECUTED)  
 2365 PATHNAME ADDRESS  
 2368 FILE TYPE  
 2376 OPEN PARM FOR ATINT FILE  
 (AT \$89B WHEN EXECUTED)  
 2377 PATHNAME ADDRESS  
 2379 I/O BUFFER AT \$1400  
 237B REFNUM=1

ProDOS Relocator -- V1.2 -- 6 SEP 86 NEXT OBJECT ADDR: 237B

ADDR DESCRIPTION/CONTENTS

```

237C ---
237D "ATINIT"

2383 ***** LOOK FOR NORMAL SYSTEM FILE *****
      (THIS CODE EXECUTES AT $8A8)

2383 $10/11 --> VOLUME DIRECTORY ENTRIES
2385 INITIALLY AT $C00
2387 OFFSET BEYOND LINKS (+4)
2389 JUMP OVER NEXT INSTRUCTION >>238D
  
```

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

```

238B PICK UP LSB
238D ---
238E BUMP BY ENTRY LENGTH (0C23)
2391 UPDATE LSB
2393 PAGE OVERFLOW? >>23A7
2395 NO, ROOM FOR ONE MORE ENTRY? (0C23)
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 >>238B
23B8 SAVE NAME LENGTH AT $280 (0280)
23BD MUST BE AT LEAST 8 CHARS LONG >>238B
23BF JUMP AROUND ERROR CODE >>23C3
23C1 ERROR - SYSTEM FILE NOT FOUND >>2430
  
```

```

23C3 ---
23C6 IS THIS ".SYSTEM"?
23C8 (SEE $24E7) (0A0C)
23CC NO, SKIP ENTRY >>238B
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 (09E2)
23E4 ADD BLANK AT END OF NAME
23E6 IN MESSAGE (09E3)
  
```

ProDOS Relocator -- V1.2 -- 6 SEP 86 NEXT OBJECT ADDR: 23EA

ADDR DESCRIPTION/CONTENTS

```

23EA NAMELEN + ERRORMSGLEN
23EC SAVE AT $24D1 (09F6)
23EF MLI: OPEN .SYSTEM FILE <BF00>
23F5 OPEN ERROR >>243D
23F7 MLI: GET EOF <BF00>
23FD CAN'T GET EOF >>243D
23FF GET HIGH BYTE (0A01)
2402 FILE TOO BIG! >>2457
2404 GET MEDIUM BYTE (0A00)
2407 MUST BE LESS THAN $9800 BYTES
2409 TOO BIG. >>2457
240B STORE LENGTH IN MLI READ PARMS (0A07)
240E GET LOW BYTE (09FF)
2411 AND STORE IN READ PARMS. (0A06)
2414 MLI: READ SYSTEM FILE INTO $2000 <BF00>
241A NO READ ERRORS >>2422
241C ERROR, BAD BUFFER?
241E YES, FILE WAS TOO LARGE >>2457
2420 ELSE, "UNABLE TO LOAD ..." >>243D
2422 MLI: CLOSE SYSTEM FILE <BF00>
2428 CLOSE ERROR >>243D
242A ENABLE MOTHERBOARD ROM (C082)
242D AND JUMP TO BEGINNING OF FILE >>2000
  
```

2430 \*\*\*\*\* ERROR HANDLERS \*\*\*\*\*

```

2430 ---
2432 PRINT "UNABLE TO FIND A .SYSTEM FILE" (0989)
243B THEN GO TO SLEEP >>2462

243D GET NAME LENGTH (09F6)
2440 LINE LENGTH
2443 LESS NAME LENGTH (09F6)
2446 DIVIDED BY 2
2447 GIVES OFFSET TO CENTER THE LINE (09F6)
244B PRINT "UNABLE TO LOAD ..." (09D1)
2455 GO TO SLEEP FOREVER >>2462
  
```

```

2457 ---
2459 PRINT "SYSTEM PROGRAM TOO LARGE" (09B1)
2462 GO TO SLEEP FOREVER >>2462
  
```

2464 \*\*\*\*\* DATA AREA \*\*\*\*\*

```

2464 '*** UNABLE TO FIND A ".SYSTEM" FILE **'
248C '*** SYSTEM PROGRAM TOO LARGE ***'
24AC '*** UNABLE TO LOAD X.SYSTEM *****'
24D1 NAME LEN +13H (LEN OF MSG)
  
```

ProDOS Relocator -- V1.2 -- 6 SEP 86      NEXT OBJECT ADDR: 24D1  
 ADDR    DESCRIPTION/CONTENTS

24D2 MLI: OPEN PARM LIST (AT \$9F7 WHEN EXECUTING)  
 24D3 PATHNAME IS AT \$280  
 24D5 I/O BUFFER AT \$1400  
 24D7 REFNUM=1  
 24D8 MLI: GET EOF PARM LIST (AT \$9FD)  
 24D9 REFNUM=1  
 24DA EOF MARK POSITION  
 24DD MLI: READ LIST (AT \$A02)  
 24DE REFNUM=1  
 24DF READ INTO \$2000  
 24E1 LENGTH (FROM EOF MARK)  
 24E3 ACTUAL LENGTH READ  
 24E5 MLI: CLOSE LIST (AT \$A0A)  
 24E6 REFNUM=0, CLOSE ALL FILES  
 24E7 '.SYSTEM'

24EE \*\*\*\*\* END OF SYSTEM FILE LOADER \*\*\*\*\*

24EF \*\*\*\*\* PAGE 3 VECTOR IMAGE \*\*\*\*\*

(INCLUDES A ROUTINE AT \$3D6 THAT IS USED  
 TO CALL A DEVICE DRIVER IN AUX HIGH RAM)  
 24EE FROM MAIN Z-PAGE, (C008)  
 24F1 GET X+1 VALUES STARTING AT \$42  
 24F3 AND PUT IN AUX Z-PAGE (C009)  
 24F6 AT SAME LOCATION.  
 THE NEXT THREE BYTES (AT \$3E3) ARE PATCHED  
 WITH A JUMP TO A DEVICE DRIVER IN AUX  
 HIGH RAM WHEN THE DRIVER IS PATCHED.  
 24FB "NO DEVICE CONNECTED" ERROR (IF NOT INSTALLED)  
 24FE BACK TO MAIN Z-PAGE (C008)  
 2501 RETURN  
 2502 ADDRESS OF MLI ROUTINE THAT CALLS  
 THE RAM-BASED DEVICE DRIVER. THIS  
 ADDRESS IS USED AS THE INSTALLED  
 DRIVER'S ADDRESS IN THE GLOBAL PAGE.  
 2508 BRK HANDLER AT \$FA59  
 250A RESET AT \$FF59  
 250C POWER UP BYTE  
 250D & VECTOR TO \$FF59 >>FF59  
 2510 CTL-Y VECTOR TO \$FF59 >>FF59  
 2513 NMI VECTOR TO \$FF59 >>FF59  
 2516 IRQ HANDLER AT \$BFEB (GLOBAL PAGE)

ProDOS Relocator -- V1.2 -- 6 SEP 86      NEXT OBJECT ADDR: 2518  
 ADDR    DESCRIPTION/CONTENTS

2518 \*\*\*\*\* SET HIGH RAM FOR READ/WRITE \*\*\*\*\*  
 2518 TWO READS OF \$C08B (C08B)  
 251E RETURN

251F \*\*\*\*\* DETERMINE MACHINE ID \*\*\*\*\*  
 \$0C=\$00..0... APPLE II  
 01..0... APPLE II+  
 10..0... APPLE Iie or IIGS  
 10..1... APPLE IIC  
 11..0... APPLE /// EMULAT.  
 ..01.... 48K RAM  
 ..10.... 64K RAM  
 ..11.... 128K RAM  
 .... ..1. 80 COL CARD  
 .... ..1.1 COMPATIBLE CLOCK

251F ASSUME NOTHING AT FIRST  
 2523 GET A ROM BYTE (FBB3)  
 2526 APPLE II?  
 2528 YES, SET BIT >>254B  
 252A NO,  
 252C APPLE IIE OR IIGS?  
 252E YES, SET BIT >>254B  
 2530 NO,  
 2532 APPLE II+?  
 2534 NO, STRANGE ROM! >>2545  
 2539 REALLY A II+?  
 253B YES >>254B  
 253F /// EMULATION MODE?  
 2543 ---  
 2544 RETURN  
 2545 OTHERWISE, UNKNOWN MACHINE  
 2547 CREATE INVALID INSTR AT \$80  
 2549 AND GO THERE >>2568  
 254B UPDATE MACHID  
 254D READ/WRITE ENABLE HIGH RAM (BANK1) <2518>  
 2552 SEE IF HIGH RAM EXISTS (D000)  
 2564 IF PRESENT, MARK IN MACHID

256B \*\*\*\*\* LOOK FOR AUXILIARY RAM \*\*\*\*\*  
 (CODE MOVED TO \$80 TO ALLOW BANK SWITCH)  
 (ENTERED WITH MACHID IN ACCUMULATOR)

256B UPDATE MACHID  
 256D IF PLUS, IS II OR II+ >>25A4  
 256F IT'S A Iie OR LATER. CHECK FOR 128K.  
 2571 BANK TO AUX MEMORY (C005)  
 2577 STORE A PATTERN AT \$C00 (0C00)  
 257A AND AT \$800 (0800)



ProDOS Relocator -- V1.2 -- 6 SEP 86      NEXT OBJECT ADDR: 2580      PRODOS Relocator -- V1.2 -- 6 SEP 86      NEXT OBJECT ADDR: 2650

-----

ADDR    DESCRIPTION/CONTENTS

2580 MAKE SURE PATTERN STAYS THERE  
 2582 IT DIDN'T! >>2592  
 2584 NOW SHIFT \$C00 TO THE LEFT (0C00)  
 2587 AND SHIFT THE ACCUM TO THE LEFT  
 2588 ARE THEY STILL THE SAME? (0C00)  
 258B NO, AUX RAM NOT THERE. >>2592  
 258D DID \$800 MOVE TOO? (0800)  
 2590 NO, SO WE HAVE FULL 128K! >>2595  
 2592 DON'T HAVE 128K  
 2595 ---  
 2596 BANK BACK TO MAIN MEMORY (C004)  
 259C ONLY 64K >>25A4  
 25A0 NO, INDICATE 128K  
 25A2 IN MACHID  
 25A4 SET UP \$A/B --> "APPLE II"  
 25A6 IN MOTHERBOARD ROM AT \$FB09  
 25A9 \$B NOW \$FB  
 25AB SOMETHING WRONG >>25AF  
 25AD \$A NOW \$09  
 25AF ---  
 25B0 RETURN TO CALLER

25B1 \*\*\*\*\* DISPLAY LOAD MESSAGE \*\*\*\*\*  
 25B1 CLICK SPEAKER (C030)  
 25B4 STORE IN MAIN MEMORY (C00C)  
 25B7 80 COL DISPLAY OFF (C000)  
 25BA SET NORMAL VIDEO <FE84>  
 25BD CALL MONITOR INITIALIZATION <FB2F>  
 25C0 SET VIDEO PR#0 <FE93>  
 25C3 SET KEYBD IN#0 <FE89>  
 25C6 OUT OF DECIMAL MODE  
 25C7 CLEAR SCREEN <FC58>  
 25CC PRINT "APPLE //" (25FA)  
 25D7 PRINT "PRODOS 8 " ETC. ON ROW 12 (2602)  
 25E2 PRINT 12 BLANKS ON ROW 14 (2620)  
 25ED PRINT "COPYRIGHT" ETC. ON ROW 24 (262C)  
 25F6 CLICK SPEAKER AGAIN (C030)  
 25F9 DONE

25FA \*\*\*\*\* DATA AREA \*\*\*\*\*  
 25FA 'APPLE //'      06-SEP-86'  
 2602 'PRODOS 8 V1.2'  
 2620 '  
 262C 'COPYRIGHT APPLE COMPUTER, INC., 1983-86'

-----

ADDR    DESCRIPTION/CONTENTS

2653 8 BYTES FOR SMARTPORT STATUS CALL  
 265B DRIVER ADDRESS  
 265D SPACES LEFT ON DEVICE LIST  
 265E SLOT 2 FLAG (0 = PRODOS STORAGE DEVICE IN SLOT 2)  
 265F \*\*\*\*\* DETERMINE SLOT CONFIGURATION \*\*\*\*\*  
 ---  
 ZERO SOME THINGS  
 2661 DEVCNT=\$FF (NO DEVICES YET) (BF31)  
 2668 ALL 14 DEVICES ARE UNASSIGNED  
 2670 FIRST CHECK SLOT 2  
 2674 IS A STORAGE DEVICE IN SLOT 2? <2898>  
 2677 IF NOT, SET A FLAG (265E)  
 267A NOW POINT TO SLOT 7  
 267E STORAGE DEVICE IN SLOT? <2898>  
 2681 NO. >>26DF  
 2683 GET \$CSEFF BYTE  
 2685 LOOKS LIKE 16 SECTOR DISK II. >>26AC  
 2687 LOOK LIKE 13 SECTOR DISK II?  
 2689 YES, DON'T USE IT. >>26DF

\*\*\*\*\* NON-DISK II STORAGE DEVICE \*\*\*\*\*  
 268B \$CFF BYTE = LOW BYTE OF DEVICE ADDRESS (265B)  
 268E CHECK BYTE AT OFFSET 7  
 2690 TO SEE IF IT'S A SMARTPORT  
 2692 NOT A SMARTPORT INTERFACE >>2697  
 2694 GO DO SMARTPORT STUFF >>2844  
 2697 ---  
 GET \$CSEFF (STATUS BYTE)  
 2699 CAN WE AT LEAST READ STATUS AND DATA?  
 269B ANTICIPATE FAILURE  
 269F CAN'T READ IT. NO SENSE USING IT. >>26DF  
 26A2 PUT LEFT NIBBLE OF STATUS BYTE IN \$12 <288D>  
 26A6 PUSH CLC, INDICATING ONE DRIVE  
 26A7 CARRY SET IF 2 OR 4 DRIVES  
 26A8 GET HIGH BYTE OF SLOT ROW  
 26AA ALWAYS BRANCH INTO DISK II PROCESSING >>26B9

\*\*\*\*\* LOOKS LIKE A DISK II \*\*\*\*\*  
 26AC \$12 = 0 FOR DISK II  
 26AF PUSH SEC ON STACK (DISK II HAS 2 DRIVES)  
 26B0 GET LOW BYTE OF DISK II DRIVER (\$00) (27D8)  
 26B3 SAVE IT IN RELOC DATA AREA (265B)  
 26B6 GET HIGH BYTE OF DISK II DRIVER (\$D0) (27D9)

ProDOS Relocator -- V1.2 -- 6 SEP 86      NEXT OBJECT ADDR: 26B6  
 ADDR    DESCRIPTION/CONTENTS

\*\*\*\*\* COMMON PROCESSING \*\*\*\*\*

26B9 SAVE DEVICE ADDRESS HIGH BYTE (265C)  
 26BC ESTABLISH DEVICE DRIVER IN GLOBAL PAGE <2814>  
 26BF ONLY ONE DRIVE?  
 26C0 YES, GO TO NEXT SLOT >>26DE  
 IF TWO DRIVES WERE ASSIGNED, MOVE THEM TO  
 THE BOTTOM OF THE LIST IN REVERSE ORDER

26DE ---  
 CARRY IS NOW CLEAR IF A PRODOS STORAGE DEVICE  
 WAS FOUND IN THIS SLOT. OTHERWISE, CARRY IS SET.  
 26DF GO MARK SLTBYT TO SHOW ROMS IN SLOT <2768>  
 26E2 MOVE DOWN ONE SLOT  
 26E8 WE'VE DONE ALL SLOTS >>26ED  
 26EA CHECK NEXT SLOT >>267E  
 26F3 STASHED ANY DEVICES AT BOTTOM OF LIST? (265D)  
 26F6 NO. >>271A  
 26F8 YES, MOVE THEM BACK  
 2700 IN REVERSE ORDER.  
 2715 DONE WHEN X=Y (265D)  
 271A ---

271C START AT BOTTOM OF SEARCH LIST (BF31)  
 271F GET A DEVICE FROM LIST (BF32)  
 2722 PUT IT ON THE STACK  
 2725 IS IT THE CURRENT SLOT? (BF30)  
 2729 NO, KEEP LOOKING >>272D  
 272B YES, TAKE IT OFF THE STACK  
 272C INDICATE CURRENT SLOT FOUND  
 272D ---  
 272E MORE TO CHECK >>271F  
 2730 GET DEVICE COUNT (BF31)  
 2734 CURRENT SLOT NOT FOUND! >>274A  
 2736 PUT CURRENT DRIVE AT (BF30)  
 2739 BOTTOM OF SEARCH LIST (BF32)  
 273D ONLY ONE DEVICE ON LIST >>2751  
 2740 ONLY ONE DRIVE ON BOOT SLOT >>274A  
 2742 CHANGE DRIVE NUMBER  
 2744 STORE OTHER DRIVE NEXT TO LAST (BF32)  
 2748 CURRENT SLOT ONLY SLOT ON LINE >>2751  
 274A GET OTHER DEVICES  
 274B MOVE THEM AHEAD OF CURRENT DRIVE (BF32)  
 274F STILL MORE TO DO >>274A

2751 DO CHECKSUM ON ROM <27EA>  
 2754 DOESN'T SAY "APPLE II"1 >>275A  
 2756 WE'RE HAPPY, STORE FINISHED MACHID (BF98)  
 2759 AND LEAVE  
 275A UNKNOWN MACHINE, SO DIE HORRIBLY! >>2545

ProDOS Relocator -- V1.2 -- 6 SEP 86      NEXT OBJECT ADDR: 275A  
 ADDR    DESCRIPTION/CONTENTS

\*\*\*\*\* PUT A DEVICE ON DEVICE LIST \*\*\*\*\*

275D  
 275D COMBINE DSSS with I111  
 2762 BUMP DEVICE COUNT BY ONE  
 2763 AND ADD DRIVE TO SYSTEM SEARCH LIST (BF32)  
 2766 ROLL LEFT ANTICIPATING ROLL RIGHT  
 2767 RETURN

2768 \*\*\*\*\* IDENTIFY I/O CARD \*\*\*\*\*  
 2768 WE ALREADY FOUND ROM IN THIS SLOT >>27C9  
 276C CHECK SIGNATURE ON CARD FOR THUNDERCLOCK  
 2771 NOT IT >>278D  
 2777 THUNDERCLOCK, WHICH SLOT?  
 2779 SAVE SLOT NUMBER (LESS 1)  
 277B IN CLOCK CODE RELOCATION TABLE (22D1)  
 2780 ENABLE CLOCK/CALENDAR JUMP IN GLOBALS (BF06)  
 2785 NO MACHIDI! >>2751  
 2787 INDICATE THAT A CLOCK IS PRESENT  
 2789 AND UPDATE MACHID  
 278B GO MARK ROM IN THIS SLOT >>27C9  
 278C CHECK FOR PASCAL 1.1 PROTOCOL

278D ---  
 2791 \$Cs05 = \$38?  
 2793 DOESN'T GET TO FIRST BASE >>27B8  
 2799 \$Cs07 = \$18?  
 279B NO. >>27B8  
 27A1 \$Cs0B = \$01?  
 27A3 NO, BAD SIGNATURE >>27B8  
 27A8 YES, GET LEFT NIBBLE  
 27AA 80 COLUMN CARD?  
 27AC NO, UNKNOWN CARD >>27B8  
 27B0 NO MACHIDI! >>2751  
 27B2 MARK 80 COLUMN CARD PRESENT  
 27B4 AND UPDATE MACHID  
 27B6 GO MARK ROM IN THIS SLOT >>27C9

27B8 UNKNOWN CARD, CHECK ROM TO  
 27BC SEE IF IT WILL HOLD A VALUE  
 27C2 FOR SOME TIME.

27C9 WE FOUND ROM IN THIS SLOT  
 27CB CONVERT SLOT NUMBER  
 27CE TO A BIT POSITION (27E2)  
 27D1 AND OR INTO SLTBYT (BF99)  
 27D7 RETURN TO CALLER

ProDOS Relocator -- V1.2 -- 6 SEP 86 NEXT OBJECT ADDR: 27D7  
 ADDR DESCRIPTION/CONTENTS

27D8 \*\*\*\*\* DATA AREA \*\*\*\*\*  
 27D8 DISK II DEVICE DRIVER ENTRY POINT  
 27DA DEVICE SIGNATURE FOR:  
 27DC +0,+2,+4,+6 = THUNDERCLOCK  
 27DE +1,+3,+5,+7 = DISK  
 27E0 (+7 NOT CHECKED)  
 27E2 BIT POSITION TABLE FOR SLOTS  
 27E5 (ALSO USED IN CHECKSUM CALCS)  
 27EA \*\*\*\*\* COMPUTE AUTOSTART ROM CHECKSUM \*\*\*\*\*  
 ---  
 27EA GET ZERO IN INDEX REGISTER (27E2)  
 27EB POINT TO \$FB09 ("APPLE II" IN ROM)  
 27E0 MAKE SURE UPPER CASE  
 27F5 UPDATE CHECKSUM (27E2)  
 27F8 PUT HIGH BIT IN CARRY (27E2)  
 27FC DO 8 BYTES IN ALL (27E5)  
 2801 ACCUM = \$08  
 2805 ACCUM = \$80  
 2807 TURN ON HIGH BIT (27E2)  
 280A ADD A FUDGE FACTOR  
 280C OH NO! 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 0SS0000 (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 (BF32)  
 2829 STORE FINAL DEVICE COUNT (BF31)  
 282C SHIFT DRIVE INDICATOR BACK TO CARRY  
 282D GET LOW BYTE OF DEVICE DRIVER ADDRESS (265B)  
 2830 PUT IN GLOBAL PAGE FOR DRIVE 1 (BF10)

ProDOS Relocator -- V1.2 -- 6 SEP 86 NEXT OBJECT ADDR: 2833  
 ADDR DESCRIPTION/CONTENTS

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 (265B)  
 2850 ADD THREE TO GET SMARTPORT ENTRY  
 2852 POKE INTO SMARTPORT CALL (285C)  
 2858 POKE THE HIGH BYTE, TOO. (285D)  
 285B SELF MODIFIED TO CALL THE SMARTPORT <0000>  
 285E WITH A STATUS COMMAND.  
 285F PARMLIST AT \$28AB  
 2861 GET NUMBER OF DEVICES ON LINE (2653)  
 2864 NONE ON LINE! >>288A  
 2866 INDICATE IF DRIVE 2 EXISTS.  
 2868 PUT DRIVER ADDRESS IN GLOBAL PAGE <2814>  
 286D IS THIS SLOT 5?  
 286F NO. >>288A  
 2871 SLOT 2 BEING USED BY STORAGE DEVICE? (265E)  
 2874 YES, TWO DRIVES IS ALL YOU GET! >>288A  
 2876 GET NUMBER OF DEVICES AGAIN (2653)  
 2879 MORE THAN TWO DRIVES?  
 287B NO. >>288A  
 287D SET CARRY IF DRIVE 4 EXISTS.  
 287F PUT THEM IN SLOT 2  
 2883 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  
 2898 \*\*\*\*\* 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 >>28AA  
 28A9 SUCCESS--THIS IS A PRODOS STORAGE DEVICE.  
 28AA RETURN

ProDOS Relocator -- V1.2 -- 6 SEP 86      NEXT OBJECT ADDR: 28AA  
 ADDR    DESCRIPTION/CONTENTS

```

28AB ***** COMMAND LIST FOR SMARTPORT CALL *****
28AB 3 PARAMETERS
28AC OVERALL STATUS CALL
28AD PUT STATUS DATA AT $2653
28AF STATUS CODE IS $00

28B0 ***** RELOCATION ROUTINE *****
      (X/Y REGS CONTAIN TABLE ADDR)

28B0 SAVE PASSED TABLE ADDRESS
28B4 ACCESS IIGS STATEREG BYTE TO (C068)
28B7 TURN OFF SLOT ROM, ENSURE ROM BANK 0
28BC ---
28BE GET OPERATION CODE
28C0 VALID OPERATION? (4 OR LESS)
28C2 NO, ERROR >>2936
28C6 $14/15 --> OUTPUT BLOCK
28D0 $16/17 --> LENGTH
28D9 NEGATIVE LENGTH? >>2938
28DB CHECK OPERATION CODE
28DC ZERO BLOCK? >>2941
28DF NO, $12/13 = $18/19 --> INPUT BLOCK
28E9 $1A/1B --> END OF INPUT BLOCK
28F6 COPY BLOCK ONLY? >>2965
28F8 SAVE RELOCATION OPERATION CODE (2A8F)
28FE SAVE NUMBER OF RANGES TO CHECK (2A90)
2902 ---
2903 COPY START PAGES TO TABLE
290E ---
290F AND END PAGES
291A ---
291B AND FINALLY, RELOCATION FACTORS
2923 BUMP TO NEXT TABLE ENTRY <296B>
2926 RESTORE OPERATION CODE (2A8F)
292B RELOCATE INSTRUCTIONS? >>293B

292D ***** 2/3 - RELOCATE ADDRESSES *****
292D NO, RELOCATE ADDRESS <29CD>
2930 COPY BLOCK <2976>
2933 AND CONTINUE IF ALL WENT WELL >>28BC
2936 NORMAL EXIT
2937 RETURN
2938 JUMP TO ERROR EXIT >>2A03
  
```

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

```

293B ***** 4 - RELOCATE INSTRUCTIONS *****
293B RELOCATE INSTRUCTIONS <29DF>
293E AND THEN COPY BLOCK >>2930

2941 ***** 0 - ZERO BLOCK *****
2941 BUMP TABLE POINTER TO NEXT ENTRY <296B>
2946 GET NUMBER OF PAGES TO DO
2948 NO FULL PAGES? >>2956
294B ZERO AN ENTIRE PAGE
2950 BUMP PAGE POINTER
2952 AND DECREMENT LENGTH
2956 GET LENGTH OF PARTIAL LAST PAGE
2958 NO PARTIAL PAGE? >>2962
295B ZERO PARTIAL PAGE TOO
2962 DONE, GET NEXT TABLE ENTRY >>28BC

2965 ***** 1 - COPY BLOCK *****
2965 BUMP TABLE POINTER <296B>
2968 AND GO COPY BLOCK >>2930

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

2976 ***** COPY BLOCK *****
---
2976 INPTR < OUTPTR? >>2987
297C NO, GREATER? >>29AA
297E MSB'S ARE EQUAL, CHECK LSB'S ALSO
2986 EXIT IF EQUAL
2987 INPTR < OUTPTR, COPY LAST PAGES FIRST
298B BUMP BOTH INPTR AND OUTPTR BY...
298D LENGTH-1 TO POINT AT LAST BYTE
2995 START WITH SHORT LAST PAGE LENGTH
2999 ---
299A COPY BYTES BACKWARDS THROUGH MEMORY
29A1 DROP ADDRESSES AND LENGTH BY 256
29A7 AND CONTINUE UNTILL FINISHED >>2999
29A9 RETURN
  
```

ProDOS Relocator -- V1.2 -- 6 SEP 86      NEXT OBJECT ADDR: 29A9  
 -----  
 ADDR    DESCRIPTION/CONTENTS  
 -----

```

29AA INPTR > OUTPTR, COPY PAGES FORWARD
29AC HOW MANY FULL PAGES LEFT?
29AE NONE? >>29BF
29B0 COPY A FULL PAGE
29B7 AND BUMP ADDRESSES
29BB DECREMENT LENGTH BY 256
29BD AND DO ALL PAGES >>29B0
29BF GET LENGTH OF LAST PAGE
29C1 EVEN PAGE BOUNDARY? >>29CC
29C3 NO, COPY SHORT LAST PAGE
29CC RETURN

29CD ***** ADDR/PAGE RELOCATE *****
29CD GET TABLE ENTRY TYPE (2A8F)
29D1 GET PAGE TO RELOCATE
29D3 RELOCATE A SINGLE ADDRESS <2A0B>
29D6 BUMP BY 1 OR 2 BYTES (2A8F)
29D9 ADVANCE POINTER <2A27>
29DC AND CONTINUE UNTIL COMPLETE >>29CD
29DE RETURN
  
```

```

29DF ***** INSTRUCTIONS RELOCATE *****
29E1 ---
29E1 GET 6502 OPCODE
29E3 COMPUTE INSTRUCTION LENGTH <2A3A>
29E6 INVALID OPCODE? >>29F9
29E8 3 BYTE INSTRUCTIONS?
29EA NO >>29F3
29EC YES, 3 BYTE ADDRESS TO CORRECT
29EE RELOCATE ADDRESS <2A0B>
29F1 AND ADVANCE BY 3 BYTES
29F3 NEXT INSTRUCTION <2A27>
29F6 CONTINUE UNTIL FINISHED >>29DF
29F8 RETURN

***** INVALID OPCODE *****
29F9 POP THE STACK
29FB RETURN WITH POINTER TO BAD INSTRUC.
29FF DIE HORRIBLY
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) (2A90)
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 >>2A10
2A1F RETURN

2A20 ---
2A21 ADD FUDGE FACTOR TO ADDRESS (2AA1)
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 OPCODE
2A3B ISOLATE LAST TWO BITS FOR LATER
2A40 USE LAST 6 BITS AS TABLE INDEX
2A42 GET BYTE WITH 4 LENGTHS IN IT (2A4F)
2A45 ---
2A46 USING TOP TWO BITS AS INDEX... >>2A4C
2A48 SHIFT DOWN THE PROPER LENGTH
2A4C AND ISOLATE IT IN A-REG
2A4E RETURN

2A4F ***** 6502 OP LENGTH TABLE *****
      EACH BYTE CONTAINS FOUR 2 BIT LENGTHS

2A4F ---
  
```

ProDOS Relocator -- V1.2 -- 6 SEP 86      NEXT OBJECT ADDR: 2A8B  
 -----  
 ADDR    DESCRIPTION/CONTENTS  
 -----

2A8F \*\*\*\*\* RELOCATION DATA \*\*\*\*\*  
 2A8F    RELOCATION CODE (3,2,1)  
 2A90    NUMBER OF RANGES  
 2A91    START OF RANGE PAGES  
 2A99    END OF RANGE PAGES +1  
 2AA1    ADDITIVE FACTORS

2AA9 \*\*\*\*\* 2AA9-2AFF NOT USED \*\*\*\*\*  
 2AA9    NOT USED  
 2AEC  
 2B00 \*\*\*\*\* SET UP RAMDRIVE IN AUXMEM \*\*\*\*\*  
       (THIS ROUTINE PUTS THE RAMDRIVE DEVICE DRIVER  
       IN MEMORY, PUTS THE ADDRESS OF THE DRIVER  
       IN THE DEVICE DRIVER ADDRESS LIST, AND  
       ADDS THE RAMDRIVE TO THE ONLINE DEVICE LIST.)

2B00    ---  
 2B02    RELOCATE RAMDRIVE CALLER NOW AT.. (2B00)  
 2B05    TO HIGH RAM AT.. (FF00)  
 2B0D    NOW PREPARE TO MOVE  
 2B0F    RAMDRIVE DEVICE DRIVER  
 2B11    INTO AUX RAM AT \$200.  
 2B14    \$3C/\$3D --> \$2C00  
 2B18    \$3E/\$3F --> \$2DFF  
 2B1D    \$42/43 --> \$200  
 2B23    COPY MAIN MEM TO AUX MEM  
 2B24    USE AUXMOVE TO COPY IT <C311>  
 2B29    SLOT 3, DRIVE 2 DEVICE DRIVER.. (BF26)  
 2B2C    IS AT \$FF00  
 2B31    BUMP DEVICE COUNT (BF31)  
 2B37    ADD DEVICE TO ONLINE DEVICE LIST  
 2B3C    RETURN

2B3D \*\*\*\*\* 2B3D-2BFF NOT USED \*\*\*\*\*  
 2B3D    ---  
 2B7D  
 2C00 \*\*\*\*\* RAMDRIVE (/RAM) DEVICE DRIVER \*\*\*\*\*  
       (COPIED TO AND RUN AT \$200 IN AUX RAM)  
       (THIS IS THE MAIN PART OF THE DEVICE DRIVER.  
       IT IS CALLED BY THE RAMDRIVE CALLER  
       WHICH IS LOCATED AT \$FF00 IN MAIN MEMORY.)

ProDOS Relocator -- V1.2 -- 6 SEP 86      NEXT OBJECT ADDR: 2C00  
 -----  
 ADDR    DESCRIPTION/CONTENTS  
 -----

2C00    SAVE THE 80STORE SETTING (C018)  
 2C04    FORCE RAM READ/WRITE (C000)  
 2C09    COPY INPUT PARAMETERS  
 2C0B    TO AUX PAGE 3. (03BD)  
 2C11    FIRST TIME IN OR FORMAT COMMAND? (03BC)  
 2C14    NO, SKIP FORMAT LOGIC >>2C4F

\*\*\*\*\* FORMAT RAMDRIVE \*\*\*\*\*

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

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

2C4F    CONVERT BLOCK NUMBER TO PAGE NUMBER (03C1)  
 2C55    THIS PAGE IN HIGH RAM?  
 2C57    YES >>2C63  
 2C59    NO, IS IT BLOCK 3? (VOLUME BIT MAP)  
 2C5B    NO >>2C60  
 2C5D    YES, DUMMY UP A PHONY BITMAP BLOCK >>038C  
 2C60    ELSE, NORMAL READ/WRITE >>0342

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

2C63    SAVE PAGE NUMBER  
 2C64    FIND IT IN MEMORY <02E5>  
 2C67    REMEMBER READ/WRITE STATUS  
 2C68    WRITING? >>2CB8  
 2C6A    GET SAVED PAGE NUMBER  
 2C6B    DOES OPERATION INVOLVE BANK1?  
 2C6D    NO, USE BANK2 >>2C73  
 2C6F    YES, FORCE IT TO SDXXX  
 2C71    AND USE BANK1 OF AUX HIGH RAM >>2C79  
 2C73    USE BANK2 OF AUX HIGH RAM (C083)  
 2C76    AND WRITE ENABLE IT (C083)

ProDOS Relocator -- V1.2 -- 6 SEP 86 NEXT OBJECT ADDR: 2C79  
 ADDR DESCRIPTION/CONTENTS

2C79 SAVE PAGE NUMBER IN BLOCK (03C1)  
 2C7C PRESERVE HIS BUFFER ADDR (03C0)  
 2C80 DURING THE FOLLOWING TRANSFER... (03BF)  
 2C83 SELECT AUX HIGH RAM (C009)  
 2C88 USE RAMDRIVE BUFFER AS AN "IN BETWEEN" (03C0)  
 2C8B AREA WHEN TRANSFERING TO/FROM AUX HIGH RAM.  
 2C8D PRETEND THAT WAS CALLER'S BUFFER (03BF)  
 2C90 AND SET UP POINTERS AGAIN 02E5>  
 2C94 COPY BLOCK TO OR FROM RAMDRIVE BUFFER  
 2C9F THEN BACK TO MAIN ZERO PAGE (C008)  
 2CA2 RESTORE CALLER'S BUFFER ADDRESS (03BF)  
 2CA9 READING OR WRITING?  
 2CAA IF WRITING, DONE >>2CB5  
 2CAC IF READING, WRITE ENABLE HIGH RAM (BANK1) (C08B)  
 2CB5 THEN EXIT >>03DE  
 2CB8 IF WRITING, COPY HIS BLOCK TO RAMDRIVE BUFFER <02BE>  
 2CBB THEN COPY RAMDRIVE BUFFER TO AUX HIGH RAM >>026A

2CBE \*\*\*\*\* COPY BLOCK IN MAIN 48K \*\*\*\*\*  
 2CBE THIS ENTRY IS FOR THE RAMDRIVE BUFFER  
 2CC0 THIS ENTRY ASSUMES AUX MEM PAGE NUMBER IN ACCUM (03C1)  
 2CC3 THIS ENTRY ASSUMES PAGE NUMBER ALREADY SET <02E5>  
 2CC6 WRITING TO RAMDISK? >>2CDB  
 2CC8 NO, WRITE TO MAIN 48K RAM (C004)  
 2CCC COPY BLOCK AUX MEM --> MAIN MEM  
 2CD7 WRITE TO AUX MEM AGAIN (C005)  
 2CDA DONE (RETURN HERE AFTER FOLLOWING JUMP)  
 2CDB ---  
 2CDD GO BACK TO MAIN MEM PART OF DRIVER (03ED)  
 2CE0 TO COPY MAIN MEM --> AUX MEM

2CE5 \*\*\*\*\* SET BUFFER AND BLOCK ADDRESSES \*\*\*\*\*  
 2CE5 GET COMMAND (03BD)  
 2CE8 READ OR WRITE?  
 2CE9 WRITE? >>2D08  
 2CEB NO, GET HIGH BYTE OF BUFFER TO BE READ (03C0)  
 2CF2 AND LOW BYTE OF BUFF ADDRESS (03BF)  
 2CF5 \$42/43 --> FIRST PAGE OF BUFFER  
 2CF7 \$40/41 --> SECOND PAGE OF BUFFER  
 2CF9 GET PAGE NUMBER (03C1)  
 2CFE \$3C/3D --> BLOCK IN RAMDRIVE  
 2D00 \$3E/3F --> SECOND PAGE OF SAME  
 2D06 ALWAYS BRANCH AROUND WRITE CODE >>2D23

ProDOS Relocator -- V1.2 -- 6 SEP 86 NEXT OBJECT ADDR: 2D06  
 ADDR DESCRIPTION/CONTENTS

2D08 WRITE, (03C0)  
 2D0F \$3C/3D --> MAIN MEMORY ADDRESS OF BUFFER TO BE WRITTEN (03BF)  
 2D12 \$3E/3F --> SECOND PAGE OF SAME  
 2D19 \$42/43 --> BLOCK IN RAMDRIVE  
 2D1B \$40/41 --> SECOND PAGE OF SAME  
 2D23 SET SECOND PAGE ADDRESSES  
 2D27 EXIT

2D28 \*\*\*\*\* SEND HIM A DUMMY BLOCK OF ZEROES \*\*\*\*\*  
 2D28 ZERO RAMDRIVE BUFFER IN CASE READING <0331>  
 2D2B COPY BETWEEN RAMDRIVE BUFFER AND HIS BUFFER <02C3>  
 2D2E AND EXIT >>03DE

2D31 \*\*\*\*\* ZERO BLOCK BUFFER \*\*\*\*\*  
 2D31 ZERO RAMDRIVE BUFFER  
 2D33 ZERO BLOCK INDICATED BY ACCUM. (03C1)  
 2D36 SET UP BUFFER POINTERS <02E5>  
 2D3A ZERO BOTH PAGES OF BLOCK  
 2D41 AND EXIT

2D42 \*\*\*\*\* READ/WRITE IN LOW 48K \*\*\*\*\*  
 2D42 BLOCK 2 (VOLUME DIRECTORY)?  
 2D44 NO >>2D4A  
 2D46 YES, CONVERT IT BLOCK 7  
 2D48 AND GO DO I/O NOW >>2D58  
 2D4A ELSE, LESS THAN BLOCK 8?  
 2D4C YES, RETURN WITH DUMMY ZERO BLOCK. >>2D28  
 2D4E START MSB AT ZERO  
 2D50 GET ORIGINAL BLOCK NUMBER  
 2D52 BLOCK \$5D THROUGH \$5F?  
 2D54 NO >>2D5B  
 2D56 YES, ADJUST TO \$D THROUGH \$F  
 2D58 AND USE \$1A00 THRU \$1FFF IN RAMDRIVE. >>0385  
 2D5B ELSE, FOR BLOCKS \$8 THRU \$5C  
 2D5C SUBTRACT 8  
 2D5E AND DIVIDE BY 17 (\$11)  
 2D64 XREG IS QUOTIENT  
 2D65 HAS TO BRANCH!! >>2D5E  
 2D68 AND AREG IS REMAINDER  
 2D69 REMAINDER OF 17  
 2D6B NO >>2D73  
 2D6D YES, EVERY 17TH BLOCK GOES  
 2D6E IN \$1000-\$1BFF AREA  
 2D6F BY ADDING 8 TO QUOTIENT  
 2D71 AND GO DO IT >>2D85  
 2D73 BUMP QUOTIENT (START AT \$2XXX)  
 2D75 SHIFT IT TO TOP NIBBLE OF BYTE

ProDOS Relocator -- V1.2 -- 6 SEP 86      NEXT OBJECT ADDR: 2D7D  
 ADDR    DESCRIPTION/CONTENTS

2D7D GOT A REMAINDER? >>2D81  
 2D7F IF SO, DECREMENT IT (NOT USING 1)  
 2D81 THEN ADD INTO TOP NIBBLE  
 2D82 TO FORM \$10 THRU \$F (03C1)  
 2D85 BLOCK#2 FOR PAGE NUMBER  
 2D86 COPY THE BLOCK <02C0>  
 2D89 THEN EXIT >>03DE

2D8C \*\*\*\*\* READ/WRITE BITMAP BLOCK \*\*\*\*\*  
 2D8C USE RAMDRIVE BUFFER (NO ACTUAL BITMAP BLOCK)  
 2D91 SET UP BUFFER POINTERS <02E5>  
 2D94 WRITING? >>2DA9  
 2D96 NO, READING - ZERO THE RAMDRIVE BUFFER <0336>  
 2D9B COPY BITMAP IMAGE TO RAMDRIVE BUFFER (03C2)  
 2DA3 COPY BLOCK BACK TO CALLER'S BUFFER <02C3>  
 2DA6 THEN EXIT >>03DE  
 2DA9 WRITING, COPY HIS BUFFER TO RAMDRIVE BUFFER <02C3>  
 2DAC SET UP BUFFER POINTERS <02E5>  
 2DB1 COPY 16 BITMAP BYTES FROM RAMDRIVE BUFFER  
 2DB3 INTO PAGE 3 BITMAP IMAGE (03C2)  
 2DB9 THEN EXIT >>03DE

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

2DDE \*\*\*\*\* EXIT TO MAIN MEMORY \*\*\*\*\*  
 2DDE WRITE ENABLE HIGH RAM (BANK1) (C08B)  
 2DE5 RESTORE \$0STORE STATUS >>2DEA  
 2DE7 \$0STORE WAS ON (C001)  
 2DEA GO AROUND MEMORY USED BY XFER >>03EF  
 2DED LOW-ORDER BYTE AND  
 2DEE HIGH-ORDER BYTE USED BY XFER ROUTINE  
 2DEF RETURN TO \$FF44 (NORMAL EXIT)

ProDOS Relocator -- V1.2 -- 6 SEP 86      NEXT OBJECT ADDR: 2DFB  
 ADDR    DESCRIPTION/CONTENTS

2DFB USE ROM XFER ROUTINE TO DO IT >>C314  
 2DFE TWO BYTES NOT USED  
 2E00 \*\*\*\*\* RAMDRIVE CALLER (RUNS AT \$FF00) \*\*\*\*\*  
 (USED TO CALL MAIN PART OF RAMDRIVE DEVICE  
 DRIVER WHICH IS AT \$200 IN AUX MEMORY.  
 ROUTINE AT \$FF65 IS USED TO TRANSFER DATA  
 FROM MAIN TO AUX MEM.)

2E00 ---  
 2E00 SAVE ZPAGE STUFF I WILL CLOBBER  
 2E03 FROM \$3C THRU \$47 (FF84)  
 2E05  
 2E0D SAVE \$3ED/E THAT XFER ROUTINE WILL CLOBBER (03ED)  
 2E16 COMMAND = STATUS?  
 2E18 IF SO, SIMPLE EXIT WILL DO >>2E44  
 2E1A ELSE, TOO BIG A COMMAND NUM?  
 2E1C IF SO, ERROR >>2E3B  
 2E1E ELSE, INVERT BITS OF CMD  
 2E20 AND SAVE IT  
 2E22 FORMAT? >>2E2C  
 2E24 NO, CHECK BLOCK NUMBER  
 2E28 MUST BE <128 FOR RAMDRIVE  
 2E2C GOING TO \$200 IN AUX MEMORY  
 FF33

2E38 USE XFER ROUTINE TO GET THERE >>C314  
 2E3B I/O ERROR RETURN CODE  
 2E3D EXIT >>2E41  
 2E3F WRITE PROTECTED RETURN CODE  
 2E41 ---  
 2E42 ERROR EXIT >>2E47  
 2E44 NORMAL EXIT, RETURN CODE IS 0  
 2E47 ---  
 2E4B RESTORE ZERO PAGE (FF84)  
 2E53 AND \$3ED/E (FF82)  
 2E56 HARMLESS INSTRUCTION MAKES SURE \$FF58 IS AN RTS (0060)  
 NOTE: THERE ARE ONLY THREE SURE THINGS IN LIFE:  
 DEATH, TAXES, AND AN RTS AT \$FF58.  
 2E64 AND EXIT TO CALLER WHEN THRU

2E65 \*\*\*\*\* COPY MAIN TO AUX BLOCK \*\*\*\*\*  
 (CALLED FROM AUX MEM HANDLER)  
 FF65  
 2E65 WRITE IN AUX 48K (C005)  
 2E6A COPY BOTH PAGES OF BLOCK  
 2E75 WRITE IN MAIN 48K AGAIN (C004)  
 2E7A GO TO \$2DA IN AUX MEMORY TO RETURN (03ED)  
 2E7F RETURN TO AUX MEM HANDLER AGAIN >>FF33



ProDOS Relocator -- V1.2 -- 6 SEP 86                   NEXT OBJECT ADDR: 2E7F

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

2E82 \*\*\*\*\* DATA AREA \*\*\*\*\*

FF82  
2E82   SAVE \$3ED,\$3EE  
FF83

FF84  
2E84   ZERO PAGE SAVE AREA

2E90 \*\*\*\*\* \$2F90-\$2EFF NOT USED \*\*\*\*\*  
          (NOTE: THE AREA FROM \$FF90-\$FF99 IS RESERVED  
          FOR THE /RAM CALLER. FROM \$FF9B TO \$FFFF  
          IS RESERVED FOR THE IRQ HANDLER.)

2E90   NOT USED

2F00 \*\*\*\*\* START OF MLI LOAD IMAGE \*\*\*\*\*

2F00   MLI LOAD IMAGE AT \$2F00

ProDOS Relocator -- V1.3 -- 2 DEC 86      NEXT OBJECT ADDR: 2000  
 ADDR    DESCRIPTION/CONTENTS

2000    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 \$2B00 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    ---  
 \*\*\*\*\* DISPLAY LOAD MESSAGE \*\*\*\*\*  
 25B1    CLICK SPEAKER (C030)  
 25B4    STORE IN MAIN MEMORY (C00C)  
 25B7    80 COL DISPLAY OFF (C000)  
 25BA    SET NORMAL VIDEO <FE84>  
 25BD    CALL MONITOR INITIALIZATION <FB2F>  
 25C0    SET VIDEO PR#0 <FE93>  
 25C3    SET KEYBD IN#0 <FE89>  
 25C6    OUT OF DECIMAL MODE  
 25C7    CLEAR SCREEN <FC58>  
 25CC    PRINT "APPLE //" (2605)  
 25D7    PRINT "PRODOS 8 " ETC. ON ROW 12 (260D)  
 25E2    PRINT 12 BLANKS ON ROW 14 (262B)  
 25ED    PRINT "COPYRIGHT" ETC. ON ROW 23 (2637)  
 25F8    PRINT "ALL RIGHTS RESERVED" ON ROW 24 (265E)  
 2601    CLICK SPEAKER AGAIN (C030)  
 2604    DONE

ProDOS Relocator -- V1.3 -- 2 DEC 86      NEXT OBJECT ADDR: 2605  
 ADDR    DESCRIPTION/CONTENTS

2605    \*\*\*\*\* DATA AREA \*\*\*\*\*  
 2605    'APPLE II'  
 260D    'PRODOS 8 V1.3                    2-DEC-86'  
 262B    '  
 2637    'COPYRIGHT APPLE COMPUTER, INC., 1983-86'  
 265E    'ALL RIGHTS RESERVED.'

2672    8 BYTES FOR SMARTPORT STATUS CALL  
 267A    DRIVER ADDRESS  
 267C    SPACES LEFT ON DEVICE LIST  
 267D    SLOT 2 FLAG (0 = PRODOS STORAGE DEVICE IN SLOT 2)

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

\*\*\*\*\* NON-DISK II STORAGE DEVICE \*\*\*\*\*  
 26AA    CSFF BYTE = LOW BYTE OF DEVICE ADDRESS (267A)  
 26AD    CHECK BYTE AT OFFSET 7  
 26AF    TO SEE IF IT'S A SMARTPORT  
 26B1    NOT A SMARTPORT INTERFACE >>26B6  
 26B3    GO DO SMARTPORT STUFF >>2863  
 26B6    ---  
 26B8    GET \$CSFE (STATUS BYTE)  
 26BA    CAN WE AT LEAST READ STATUS AND DATA?  
 26BE    ANTICIPATE FAILURE  
 26BF    CAN'T READ IT. NO SENSE USING IT. >>26FE  
 26C1    PUT LEFT NIBBLE OF STATUS BYTE IN \$12 <28C9>  
 26C5    PUSH CLC, INDICATING ONE DRIVE  
 26C6    CARRY SET IF 2 OR 4 DRIVES  
 26C7    GET HIGH BYTE OF SLOT ROM  
 26C9    ALWAYS BRANCH INTO DISK II PROCESSING >>26D8

ProDOS Relocator -- V1.3 -- 2 DEC 86 NEXT OBJECT ADDR: 26C9  
 ADDR DESCRIPTION/CONTENTS

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

26CB \$12 = 0 FOR DISK II  
 26CE PUSH SEC ON STACK (DISK II HAS 2 DRIVES)  
 26CF GET LOW BYTE OF DISK II DRIVER (\$00) (27F7)  
 26D2 SAVE IT IN RELOC DATA AREA (267A)  
 26D5 GET HIGH BYTE OF DISK II DRIVER (\$D0) (27F8)

\*\*\*\*\* COMMON PROCESSING \*\*\*\*\*

26DB ESTABLISH DEVICE DRIVER IN GLOBAL PAGE <2833>  
 26DE ONLY ONE DRIVE?  
 26DF YES, GO TO NEXT SLOT >>26FD  
 IF TWO DRIVES WERE ASSIGNED, MOVE THEM TO THE BOTTOM OF THE LIST IN REVERSE ORDER

---  
 CARRY IS NOW CLEAR IF A PRODOS STORAGE DEVICE WAS FOUND IN THIS SLOT. OTHERWISE, CARRY IS SET.  
 GO MARK SLTBYT TO SHOW ROMS IN SLOT <2787>

2701 MOVE DOWN ONE SLOT  
 2707 WE'VE DONE ALL SLOTS >>270C  
 2709 CHECK NEXT SLOT >>269D  
 2712 STASHED ANY DEVICES AT BOTTOM OF LIST? (267C)  
 2715 NO. >>2739  
 2717 YES, MOVE THEM BACK  
 271F IN REVERSE ORDER.  
 2734 DONE WHEN X=Y (267C)

---  
 2739 START AT BOTTOM OF SEARCH LIST (BF31)  
 273B GET A DEVICE FROM LIST (BF32)  
 2741 PUT IT ON THE STACK  
 2744 IS IT THE CURRENT SLOT? (BF30)  
 2748 NO, KEEP LOOKING >>274C  
 274A YES, TAKE IT OFF THE STACK  
 274B INDICATE CURRENT SLOT FOUND  
 274C

---  
 274D MORE TO CHECK >>273E  
 274F GET DEVICE COUNT (BF31)  
 2753 CURRENT SLOT NOT FOUND! >>2769  
 2755 PUT CURRENT DRIVE AT (BF30)  
 2758 BOTTOM OF SEARCH LIST (BF32)  
 275C ONLY ONE DEVICE ON LIST >>2770  
 275F ONLY ONE DRIVE ON BOOT SLOT >>2769  
 CHANGE DRIVE NUMBER  
 2761 STORE OTHER DRIVE NEXT TO LAST (BF32)  
 2763 CURRENT SLOT ONLY SLOT ON LINE >>2770  
 2767 GET OTHER DEVICES  
 276A MOVE THEM AHEAD OF CURRENT DRIVE (BF32)  
 276E STILL MORE TO DO >>2769

ProDOS Relocator -- V1.3 -- 2 DEC 86 NEXT OBJECT ADDR: 276E  
 ADDR DESCRIPTION/CONTENTS

2770 DO CHECKSUM ON ROM <2809>  
 2773 DOESN'T SAY "APPLE II"! >>2779  
 2775 WE'RE HAPPY, STORE FINISHED MACHID (BF98)  
 2778 AND LEAVE  
 2779 UNKNOWN MACHINE, SO DIE HORRIBLY! >>2545

277C \*\*\*\*\* PUT A DEVICE ON DEVICE LIST \*\*\*\*\*

277C COMBINE DSSS with IIII  
 2781 BUMP DEVICE COUNT BY ONE  
 2782 AND ADD DRIVE TO SYSTEM SEARCH LIST (BF32)  
 2785 ROLL LEFT ANTICIPATING ROLL RIGHT  
 2786 RETURN

2787 \*\*\*\*\* IDENTIFY I/O CARD \*\*\*\*\*

2787 WE ALREADY FOUND ROM IN THIS SLOT >>27E8  
 278B CHECK SIGNATURE ON CARD FOR THUNDERCLOCK  
 2790 NOT IT >>27AC  
 2796 THUNDERCLOCK, WHICH SLOT?  
 2798 SAVE SLOT NUMBER (LESS 1)  
 279A IN CLOCK CODE RELOCATION TABLE (22D1)  
 279F ENABLE CLOCK/CALENDAR JUMP IN GLOBALS (BF06)  
 27A4 NO MACHID! >>2770  
 27A6 INDICATE THAT A CLOCK IS PRESENT  
 27A8 AND UPDATE MACHID  
 27AA GO MARK ROM IN THIS SLOT >>27E8  
 CHECK FOR PASCAL 1.1 PROTOCOL

---  
 27AC \$CS05 = \$38?  
 27B0 DOESN'T GET TO FIRST BASE >>27D7  
 27B8 \$CS07 = \$18?  
 27BA NO. >>27D7  
 27C0 \$CS0B = \$01?  
 27C2 NO, BAD SIGNATURE >>27D7  
 27C7 YES, GET LEFT NIBBLE  
 27C9 80 COLUMN CARD?  
 27CB NO, UNKNOWN CARD >>27D7  
 27CF NO MACHID! >>2770  
 27D1 MARK 80 COLUMN CARD PRESENT  
 27D3 AND UPDATE MACHID  
 27D5 GO MARK ROM IN THIS SLOT >>27E8  
 27D7 UNKNOWN CARD, CHECK ROM TO  
 27DB SEE IF IT WILL HOLD A VALUE  
 27E1 FOR SOME TIME.

ProDOS Relocator -- V1.3 -- 2 DEC 86 NEXT OBJECT ADDR: 27E6

ADDR DESCRIPTION/CONTENTS

27E8 WE FOUND ROM IN THIS SLOT  
 27EA CONVERT SLOT NUMBER  
 27ED TO A BIT POSITION (2801)  
 27F0 AND OR INTO SLTBYT (BF99)  
 27F6 RETURN TO CALLER

27F7 \*\*\*\*\* DATA AREA \*\*\*\*\*

27F7 DISK II DEVICE DRIVER ENTRY POINT

27F9 DEVICE SIGNATURE FOR:  
 27FB +0,+2,+4,+6 = THUNDERCLOCK  
 27FD +1,+3,+5,+7 = DISK  
 27FF (+7 NOT CHECKED)

2801 BIT POSITION TABLE FOR SLOTS  
 2804 (ALSO USED IN CHECKSUM CALCS)

2809 \*\*\*\*\* COMPUTE AUTOSTART ROM CHECKSUM \*\*\*\*\*

2809 ---  
 280A GET ZERO IN INDEX REGISTER (2801)  
 280D POINT TO \$FB09 ("APPLE II" IN ROM)  
 280F MAKE SURE UPPER CASE  
 2814 UPDATE CHECKSUM (2801)  
 2817 PUT HIGH BIT IN CARRY (2801)  
 281B DO 8 BYTES IN ALL (2804)  
 2820 ACCUM = \$08  
 2824 ACCUM = \$80  
 2826 TURN ON HIGH BIT (2801)  
 2829 ADD A FUDGE FACTOR  
 282B OH NO! A CLONE! >>2830  
 282D PASSED THE TEST...RETURN WITH MACHID  
 282F RETURN  
 2830 ELSE, RETURN WITH ZERO MACHID  
 2832 RETURN

2833 \*\*\*DEVICE DRIVER IN GLOBAL PAGE \*\*\*\*\*

2833 SAVE CARRY (NUMBER OF DRIVES)  
 2834 GET HIGH BYTE OF SLOT ADDRESS  
 2836 MAKE IT SLOT NUMBER  
 2838 TIMES 2  
 2839 USE LATER IN Y-REG  
 283A NOW GET SLOT\*16 IN ACCUM  
 283C NOW HAVE 0SS0000 (DRIVE 1)  
 283D PUT DEVICE ID ON DEVICE LIST <277C>  
 2840 GET BACK CARRY (NUMBER OF DRIVES)  
 2841 ROLL CARRY INTO ACCUM  
 2842 ONLY ONE DRIVE. >>2848

ProDOS Relocator -- V1.3 -- 2 DEC 86 NEXT OBJECT ADDR: 2844

ADDR DESCRIPTION/CONTENTS

2844 TWO DRIVES. BUMP DEVICE COUNT  
 2845 AND PUT SECOND DRIVE ON SEARCH LIST (BF32)  
 2848 STORE FINAL DEVICE COUNT (BF31)  
 284B SHIFT DRIVE INDICATOR BACK TO CARRY  
 284C GET LOW BYTE OF DEVICE DRIVER ADDRESS (267A)  
 284F PUT IN GLOBAL PAGE FOR DRIVE 1 (BF10)  
 2852 ONLY ONE DRIVE >>2857  
 2854 PUT IN GLOBAL PAGE FOR DRIVE 2 (BF20)  
 2857 GET HIGH BYTE OF DEVICE DRIVER ADDRESS (267B)  
 285A PUT IN GLOBAL PAGE FOR DRIVE 1 (BF11)  
 285D ONLY ONE DRIVE >>2862  
 285F PUT IN GLOBAL PAGE FOR DRIVE 2 (BF21)  
 2862 RETURN

2863 \*\*\*\*\* HANDLE SMART PORT \*\*\*\*\*

2863 PUT LEFT NIBBLE OF STATUS BYTE IN \$12 <28C9>  
 2866 GET HIGH BYTE OF SLOT ROM ADDRESS  
 2868 STORE IT IN RELOC DATA AREA (267B)  
 286B GET PRODOS ENTRY, LOW BYTE (267A)  
 286E POKE INTO PRODOS CALL (2895)  
 2872 ADD THREE TO GET SMARTPORT ENTRY  
 2874 POKE INTO SMARTPORT CALL (2898)  
 287A POKE IN HIGH BYTE TO SMARTPORT CALL (2899)  
 287D AND TO PRODOS CALL (2896)  
 2880 CONVERT HIGH BYTE TO UNIT NUMBER  
 2884 STORE UNIT NUMBER  
 2886 PRODOS STATUS CALL  
 2888 STORE AS COMMAND CODE  
 288C ALSO ZERO BLOCK NUMBER  
 2890 SET BUFFER ADDRESS SET TO \$1000  
 2892 JUST IN CASE IT'S NEEDED.  
 2894 SELF-MODIFIED TO CALL PRODOS DEVICE DRIVER. <0000>  
 2897 SELF-MODIFIED TO CALL THE SMARTPORT <0000>  
 289B PARMLIST AT \$28AB  
 289D GET NUMBER OF DEVICES ON LINE (2672)  
 28A0 NONE ON LINE! >>28C6  
 28A2 INDICATE IF DRIVE 2 EXISTS.  
 28A4 PUT DRIVER ADDRESS IN GLOBAL PAGE <2833>  
 28A9 IS THIS SLOT 5?  
 28AB NO. >>28C6  
 28AD SLOT 2 BEING USED BY A STORAGE DEVICE? (267D)  
 28B0 YES, TWO DRIVES IS ALL YOU GET! >>28C6  
 28B2 GET NUMBER OF DEVICES AGAIN (2672)  
 28B5 MORE THAN TWO DRIVES?  
 28B7 NO. >>28C6  
 28B9 SET CARRY IF DRIVE 4 EXISTS.  
 28BB PUT THEM IN SLOT 2  
 28BF PUT DRIVER ADDRESS IN GLOBAL PAGE <2833>  
 28C6 GO PROCESS NEXT SLOT >>26FD

ProDOS Relocator -- V1.3 -- 2 DEC 86 NEXT OBJECT ADDR: 28C6  
 ADDR DESCRIPTION/CONTENTS

28C9 \*\*\*\*\* CONVERT STATUS FOR ID BYTE \*\*\*\*\*  
 28C9 GET STATUS BYTE  
 28CD SHIFT LEFT NIBBLE TO RIGHT NIBBLE  
 28D1 PUT IT IN \$12  
 28D3 RETURN  
 28D4 \*\*\*\*\* CHECK FOR PRODOS STORAGE DEVICE \*\*\*\*\*  
 28D4 RESET I/O CARD ROMS (CFFF)  
 28D9 CHECK 3 BYTES ON CONTROLLER ROM  
 28DE ANTICIPATE FAILURE  
 28DF NOT A PRODOS STORAGE DEVICE >>28E6  
 28E5 SUCCESS--THIS IS A PRODOS STORAGE DEVICE.  
 28E6 RETURN

28E7 \*\*\*\*\* COMMAND LIST FOR SMARTPORT CALL \*\*\*\*\*  
 28E7 3 PARAMETERS  
 28E8 OVERALL STATUS CALL  
 28E9 PUT STATUS DATA AT \$2653  
 28EB STATUS CODE IS \$00  
 28EC \*\*\*\*\* RELOCATION ROUTINE \*\*\*\*\*  
 (X/Y REGS CONTAIN TABLE ADDR)

28EC SAVE PASSED TABLE ADDRESS  
 28F0 ACCESS IIGS STATEREG BYTE TO (C068)  
 28F3 TURN OFF SLOT ROM, ENSURE ROM BANK 0  
 28F8 ---  
 28FA GET OPERATION CODE  
 28FC VALID OPERATION? (4 OR LESS)  
 28FE NO, ERROR >>2972  
 2902 \$14/15 --> OUTPUT BLOCK  
 290C \$16/17 --> LENGTH  
 2915 NEGATIVE LENGTH? >>2974  
 2917 CHECK OPERATION CODE  
 2918 ZERO BLOCK? >>297D  
 291B NO, \$12/13 = \$18/19 --> INPUT BLOCK  
 2925 \$1A/1B --> END OF INPUT BLOCK  
 2932 COPY BLOCK ONLY? >>29A1  
 2934 SAVE RELOCATION OPERATION CODE (2ACB)  
 293A SAVE NUMBER OF RANGES TO CHECK (2ACC)  
 293E ---  
 293F COPY START PAGES TO TABLE  
 294A --- AND END PAGES  
 294B ---  
 2956 ---  
 2957 AND FINALLY, RELOCATION FACTORS  
 295F BUMP TO NEXT TABLE ENTRY <29A7>

ProDOS Relocator -- V1.3 -- 2 DEC 86 NEXT OBJECT ADDR: 2962  
 ADDR DESCRIPTION/CONTENTS

2962 RESTORE OPERATION CODE (2ACB)  
 2967 RELOCATE INSTRUCTIONS? >>2977  
 2969 \*\*\*\*\* 2/3 - RELOCATE ADDRESSES \*\*\*\*\*  
 2969 NO, RELOCATE ADDRESS <2A09>  
 296C COPY BLOCK <29B2>  
 296F AND CONTINUE IF ALL WENT WELL >>28F8  
 2972 NORMAL EXIT  
 2973 RETURN  
 2974 JUMP TO ERROR EXIT >>2A3F  
 2977 \*\*\*\*\* 4 - RELOCATE INSTRUCTIONS \*\*\*\*\*  
 2977 RELOCATE INSTRUCTIONS <2A1B>  
 297A AND THEN COPY BLOCK >>296C

297D \*\*\*\*\* 0 - ZERO BLOCK \*\*\*\*\*  
 297D BUMP TABLE POINTER TO NEXT ENTRY <29A7>  
 2982 GET NUMBER OF PAGES TO DO  
 2984 NO FULL PAGES? >>2992  
 2987 ZERO AN ENTIRE PAGE  
 298C BUMP PAGE POINTER  
 298E AND DECREMENT LENGTH  
 2992 GET LENGTH OF PARTIAL LAST PAGE  
 2994 NO PARTIAL PAGE? >>299E  
 2997 ZERO PARTIAL PAGE TOO  
 299E DONE, GET NEXT TABLE ENTRY >>28F8

29A1 \*\*\*\*\* 1 - COPY BLOCK \*\*\*\*\*  
 29A1 BUMP TABLE POINTER <29A7>  
 29A4 AND GO COPY BLOCK >>296C  
 29A7 \*\*\*\*\* ADVANCE TABLE POINTER \*\*\*\*\*  
 29A7 ADD FINAL ENTRY INDEX..  
 29AB TO TABLE ENTRY ADDRESS  
 29B1 RETURN

29B2 \*\*\*\*\* COPY BLOCK \*\*\*\*\*  
 29B2 ---  
 29B6 INPTR < OUTPTR? >>29C3  
 29B8 NO, GREATER? >>29E6  
 29BA MSB'S ARE EQUAL, CHECK LSB'S ALSO  
 29C2 EXIT IF EQUAL  
 29C3 INPTR < OUTPTR, COPY LAST PAGES FIRST  
 29C7 BUMP BOTH INPTR AND OUTPTR BY...  
 29C9 LENGTH-1 TO POINT AT LAST BYTE

ProDOS Relocator -- V1.3 -- 2 DEC 86      NEXT OBJECT ADDR: 29D1  
 ADDR    DESCRIPTION/CONTENTS

29D1    START WITH SHORT LAST PAGE LENGTH  
 29D5    ---  
 29D6    COPY BYTES BACKWARDS THROUGH MEMORY  
 29DD    DROP ADDRESSES AND LENGTH BY 256  
 29E3    AND CONTINUE UNTIL FINISHED >>29D5  
 29E5    RETURN  
 29E6    INPTR > OUTFTR, COPY PAGES FORWARD  
 29E8    HOW MANY FULL PAGES LEFT?  
 29EA    NONE? >>29FB  
 29EC    COPY A FULL PAGE  
 29F3    AND BUMP ADDRESSES  
 29F7    DECREMENT LENGTH BY 256  
 29F9    AND DO ALL PAGES >>29FC  
 29FB    GET LENGTH OF LAST PAGE  
 29FD    EVEN PAGE BOUNDARY? >>2A08  
 29FE    NO, COPY SHORT LAST PAGE  
 2A08    RETURN

2A09    \*\*\*\*\* ADDR/PAGE RELOCATE \*\*\*\*\*  
 2A09    GET TABLE ENTRY TYPE (2ACB)  
 2A0D    GET PAGE TO RELOCATE  
 2A0F    RELOCATE A SINGLE ADDRESS <2A47>  
 2A12    BUMP BY 1 OR 2 BYTES (2ACB)  
 2A15    ADVANCE POINTER <2A63>  
 2A18    AND CONTINUE UNTIL COMPLETE >>2A09  
 2A1A    RETURN  
 2A1B    \*\*\*\*\* INSTRUCTIONS RELOCATE \*\*\*\*\*

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

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

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

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 TWO BITS FOR LATER  
 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  
 2A88    AND ISOLATE IT IN A-REG  
 2A8A    RETURN

2A8B    \*\*\*\*\* 6502 OP LENGTH TABLE \*\*\*\*\*  
 EACH BYTE CONTAINS FOUR 2 BIT LENGTHS

ProDOS Relocator -- V1.3 -- 2 DEC 86                   NEXT OBJECT ADDR: 2A8B  
-----  
ADDR   DESCRIPTION/CONTENTS  
-----

2A8B   ---  
2ACB \*\*\*\*\* RELOCATION DATA \*\*\*\*\*  
2ACB   RELOCATION CODE (3,2,1)  
2ACC   NUMBER OF RANGES  
2ACD   START OF RANGE PAGES  
2AD0     
2AD5   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

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

D700 MODULE STARTING ADDRESS  
 \*\*\*\*\*  
 \*  
 \* PRODOS MACHINE LANGUAGE INTERFACE \*  
 \* THIS CODE IS MOVED INTO HIGH \*  
 \* RAM (\$DE00-\$FEFF) 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 --  
 Command  
 0042 Unit Number  
 0043 Buffer Pointer  
 0044  
 0045 Block Number  
 0046  
 0047  
 -----  
 0048 I/O Pointer - Index Block or..  
 0048 pointer into \$F600 work buffer or..  
 0048 caller's pathname buffer pointer  
 0049  
 004A I/O Pointer - Data Block  
 004B  
 004C I/O Pointer - Data Block  
 004D  
 004E I/O Pointer - Caller's Data or..  
 004E buffer pointer passed in parmlist or..  
 004E old I/O buffer  
 004F

D700 \*\*\*\*\* MLI ERROR CODES \*\*\*\*\*  
 0000 No Error  
 0001 Bad call type  
 0004 Bad parameter count  
 0025 Interrupt Table full  
 0027 I/O Error  
 0028 No device connected  
 002B Write protected

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

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

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  
 07F1  
 07F2  
 07F3  
 07F4  
 07F5  
 07F6  
 07F7  
 07F8 Slot in use

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

2278 Flag=1 when running on a IIGS

D700 \*\*\*\*\* SYSTEM GLOBAL PAGE EQUATES \*\*\*\*\*  
 BF00 Jump to MLI entry point  
 BF03 JSPARE (Jump to \$EECF, 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



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

NEXT OBJECT ADDR: D700

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

NEXT OBJECT ADDR: D700

ADDR DESCRIPTION/CONTENTS

ADDR DESCRIPTION/CONTENTS

BF31 Count (-1) active devices  
 BF32 List of active devices by DEVID  
 BF58 Memory BITMAP for low 48K  
 BF70 Open file 1 buffer address  
 BF7E Open file 8 buffer address  
 BF80 Interrupt handler 1  
 BF82 Interrupt handler 2  
 BF84 Interrupt handler 3  
 BF86 Interrupt handler 4  
 BF88 A reg save during interrupt  
 BF89 X reg save during interrupt  
 BF8A Y reg save during interrupt  
 BF8B S reg save during interrupt  
 BF8C P reg save during interrupt  
 BF8E Interrupt return address  
 BF90 Date/Time  
 BF94 File open LEVEL  
 BF95 Backup bit  
 BF96 Temporary storage  
 BF9A Prefix flag (0 = no prefix)  
 BF9B MLI active flag  
 BF9C Last MLI call return address  
 BF9E MLI X reg savearea  
 BF9F MLI Y reg savearea  
 BFA0 HIGH RAM entry/exit routines  
 BFD0 Interrupt entry/exit routines  
 BFF4 Bank switch saved state (\$E000 byte)

D700 pathname buffer  
 D800 \*\*\*\*\* FILE CONTROL BLOCKS \*\*\*\*\* starts here..  
 D800 File Control Block (FCB0) starts here..  
 D800 Reference Number  
 THE FOLLOWING 6 BYTES ARE THE FILE ID  
 D801 Device Number  
 D802 Dir Block HDR for Dir describing this File  
 D804 Dir Block containing entry itself  
 D806 File entry # in this Directory  
 D807 Storage Type  
 Flags  
 LXXX XXXX Index Block Buffer Changed  
 XLXX XXXX Data Block Buffer Changed  
 XXIX XXXX Unused  
 XXXI XXXX Directory entry needs update  
 XXXX LXXX Storage Type Changed  
 XXXX XLXX Allocate new Master Index Block  
 XXXX XXXI Allocate new Sub Index Block  
 D808 Access Byte  
 D809 Newline Character  
 D80A Buffer Number (REF NUM \* 2)  
 D80C Master Index/Key Block Number  
 D80E Current Index Block  
 D810 Current Data Block  
 D812 Mark  
 D815 End of File  
 D818 Blocks Used  
 D81A not used  
 D81B Level  
 D81C Flag - Write occurred if MSB on  
 D81D not used  
 D81F Newline Enable Mask  
 D820 FCBI through FCB7

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

C00C Reset 80 column mode  
 C029 IIGS NEWVIDEO register  
 C051 Set TEXT mode  
 C053 Set Mixed text/graphics  
 C054 Display Primary page  
 C056 Set LORES graphics mode  
 C083 Read/Write RAM 2nd 4K Bank  
 C08B Read/Write RAM 1st 4K Bank  
 C0FF Reset alternate I/O ROMs

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

| L1 | NAME1 | L2 | NAME2 | .. | 00  
 Prefix is at top of buffer such that a  
 negative index may be used to use it,  
 wrapping around to the pathname again.

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

D900 Volume Control Block (VCB0) starts here..  
 D901 Length (0000LLLL)  
 D910 File Name (Max 15)  
 D910 Unit Number  
 D911 Files Open Flag (if \$\$\$)  
 D912 Total Blocks  
 D914 Blocks Free  
 D916 Block Number of Vol Dir Key Block

ProDOS MLI -- V1.2 -- 6 SEP 86      NEXT OBJECT ADDR: D918  
 ADDR    DESCRIPTION/CONTENTS

D918 not used  
 D919 not used  
 D91A Bit Map Pointer  
       Block offset into multi-block bitmap of  
 D91C next free bit.  
 D91E Count of open files  
 D920 VCBI through VCB7  
 DA00 \*\*\*\*\* BITMAP BUFFER \*\*\*\*\*  
 DA00 Buffer 1st half  
 DB00 Buffer 2nd half  
 DC00 \*\*\*\*\* PRIMARY BUFFER \*\*\*\*\*  
       (Used for several things. DIRECTORY block offsets are  
       mapped into it below)

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

DE00 \*\*\*\*\* MLI MAIN ENTRY POINT \*\*\*\*\*  
 DE00 Clear decimal mode  
 DE01 Retrieve status byte from stack  
 DE02 and store it in global page. (BF96)  
 DE05 Save Registers (BF9F)  
 DE0B Set (\$40) -> Address of function code -1  
 DE0F Set CMDADR -> True return address  
 DE1C Retrieve status byte, (BF96)  
 DE1F push it onto the stack,  
 DE20 and pull it into status register.  
 DE24 Init Global Page System error to 0 (BF0F)  
 DE28 Get Function Code

ProDOS MLI -- V1.2 -- 6 SEP 86      NEXT OBJECT ADDR: DE2B  
 ADDR    DESCRIPTION/CONTENTS

DE2B Build hash index into Command Table (X reg)  
 DE34 Is this function code valid?  
 DE39 No >>DEB1  
 DE3C Set (\$40) -> Parameter list  
 DE49 Get parameter count required (FD4D)  
 DE4C None? >>DE6A  
 DE4E Is parameter count correct?  
 DE50 No >>DEB5  
 DE52 Check class of function (FD2D)  
 DE55 Quit?  
 DE57 Yes >>DE67  
 DE59 no.  
 DE5A \$8X - Calls to I/O Drivers >>DE70  
 DE5C \$CX/DX - Non System calls >>DE7B  
 DE5E Else, \$4X - Interrupt support  
 DE5F Isolate type (0=ALLOC, 1=DEALLOC, 2=SPECIAL)  
 DE61 Call Interrupt Support <DEFD>  
 DE64 Then Exit to Caller >>DE82  
 DE67 Go to quit code via global page >>BF03

DE6A \*\*\*\*\* MLI GET TIME CALL \*\*\*\*\*  
       \*\*\*\*\* MLI GET TIME CALL \*\*\*\*\*  
 DE6A Call Date/Time driver <BF06>  
 DE6D and exit to caller >>DE82

DE70 \*\*\*\*\* MLI READ BLOCK CALL \*\*\*\*\*  
       \*\*\*\*\* MLI READ BLOCK CALL \*\*\*\*\*  
       \*\*\*\*\* MLI WRITE BLOCK CALL \*\*\*\*\*  
       \*\*\*\*\* MLI WRITE BLOCK CALL \*\*\*\*\*  
       \$80 - Read Block  
       \$81 - Write Block

DE70 ---  
 DE71 Set \$42 -> 1 for READ, 2 for WRITE  
 DE75 Do Block I/O <DEBC>  
 DE78 Then Exit to Caller >>DE82

DE7B \*\*\*\*\* \$CX and \$DX CALLS \*\*\*\*\*  
 DE7B ---  
 DE7C Isolate function Index  
 DE7F Perform function and exit to caller <E03E>  
 DE82 \*\*\*\*\* EXIT TO CALLER \*\*\*\*\*

ProDOS MLI -- V1.2 -- 6 SEP 86                      NEXT OBJECT ADDR: DE82  
 ADDR    DESCRIPTION/CONTENTS

ProDOS MLI -- V1.2 -- 6 SEP 86                      NEXT OBJECT ADDR: DEE1  
 ADDR    DESCRIPTION/CONTENTS

```

DE82 Clear Backup
DE8A Error occurred?
DE8D Save test results
DE8E Disable interrupts
DE8F Roll out most recent "active" bit (BF9B)
DE92 Get test results back
DE93 Store in X reg
DE94 Set up Return Address on stack (BF9D)
DE9C Put test results on stack
DE9E Put error code in A reg
DE9F Restore X reg (BF9E)
DEA2 Restore Y reg (BF9F)
DEA5 Put error code on stack
DEA6 Get RAM/ROM orientation (BFF4)
DEA9 Exit via RAM Global Page >>BFA0

DEAC ***** NO DEVICE CONNECTED *****
DEAC ---
DEAE Call System Error Handler (Global Page) <BF09>

DEB1 ***** BAD SYSTEM CALL NUMBER *****
DEB1 ---
DEB3 Branch always taken >>DEB7

DEB5 ***** BAD PARAMETER COUNT *****
DEB5 ---
DEB7 Call System Error Handler <DEE1>
DEBA Exit to Caller >>DEB2

DEBC ***** BLOCK I/O SETUP *****
DEBC ---
DEBE Save Old Processor Flags
DEBF Disable Interrupts
DEC0 Copy Parameters to $43-$47
DEC8 Save Starting Buffer Page in $4F
DECD Find last page + 1
DED0 Round up if Buffer not page aligned >>DED3
DED3 Is this Memory already in use? <FC63>
DED6 Yes, then exit with error >>DEE0
DED8 No, do Block I/O <DEE4>
DEDB Error? >>DEE0
DEDD No, then exit normally
DEFD 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
DEF1 Put Device Handler Address in Jump Vector (FEBD)
DEFA Exit through Device Handler >>FEBD

DEFD ***** Interrupt Handler *****
DEFD ALLOC/DEALLOC

DEFD Save Call Type
DEFF Install unclaimed interrupt handler?
DF01 No, normal ALLOC/DEALLOC >>DF09
DF03 Yes, install a handler for unclaimed interrupts. <FD23>
DF06 Error? >>DF35
DF08 No, done.
DF09 Test bit 0
DF0A 1=DEALLOC >>DF38

DEFD ALLOC

DF0C ---
DF0E 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
DF28 Exit
DF29 Skip this Vector
DF2B Last one?
DF2D No, check another >>DF0E
DF2F Yes, Table Full Error
DF31 Always taken >>DF35
DF33 Bad Parameter Error
DF35 Call System Error Handler <BF09>

DEFD DEALLOC

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
  
```

ProDOS MLI -- V1.2 -- 6 SEP 86      NEXT OBJECT ADDR: DF4D  
 ADDR    DESCRIPTION/CONTENTS

DF4E \*\*\*\*\* IRQ Handler \*\*\*\*\*  
 DF50 ---  
 DF51 Save A reg from Monitor (BF88)  
 DF53 And X,Y,S and P (BF89)  
 DF5D Is this ROM enhanced? (DFE1)  
 DF60 Yes, skip three pulls >>DF6E  
 DF67 And RTI Address (BF8E)  
 DF6E Replace stack to original condition  
 DF72 Save active slot index (DFE7)  
 DF75 In bottom half of stack?  
 DF78 Yes, pop off 16 bytes and save them  
 DF7A ---  
 DF81 Save \$FA - \$FF (top of zero page)  
 DF83 ---  
 DF8B Is there a User Vector #1 (BF81)  
 DF8E No >>DF95  
 DF90 Yes, call it <DFE3>  
 DF93 His interrupt? >>DFBD  
 DF95 Is there a User Vector #2 (BF83)  
 DF98 No >>DF9F  
 DF9A Yes, call it <DFE6>  
 DF9D His interrupt? >>DFBD  
 DF9F Is there a User Vector #3 (BF85)  
 DFA2 No >>DFA9  
 DFA4 Yes, call it <DFE9>  
 DFA7 His interrupt? >>DFBD  
 DFA9 Is there a User Vector #4 (BF87)  
 DFAC No, didn't find service routine. >>DFE3  
 DFAE Yes, call it <DFFC>  
 DFB1 His interrupt? >>DFBD  
 DFB3 Allow 256 tries, (DFE2)  
 DFB8 then indicate error type 1 and  
 DFBA call System Death Handler. <BF0C>  
 DFBF Interrupt Serviced  
 DFC7 Restore zero page (FDBD)  
 DFD7 Is this enhanced ROM? (DFE1)  
 DFDA Yes, skip some stuff we used to have to do >>DFEE  
 DFDC Reload X and Y (BF8A)  
 DFE2 Disable I/O ROMS (CFFF)  
 DFE5 Replace active slot number (C100)  
 DFE6 Exit from Interrupt >>BFD0  
 DFF1 ENHANCE FLAG. Set to 1 by RELOCATOR if new type ROM found.  
 (That is, if ROM IRQ Vector jumps below \$D000)  
 DFF2 Unclaimed IRQ Count. Incremented when an interrupt  
 is unclaimed (256 tries are allowed).

ProDOS MLI -- V1.2 -- 6 SEP 86      NEXT OBJECT ADDR: DFF3  
 ADDR    DESCRIPTION/CONTENTS

DFE3 User Interrupt Handler #1 >>BF80  
 DFF6 User Interrupt Handler #2 >>BF82  
 DFF9 User Interrupt Handler #3 >>BF84  
 DFFC User Interrupt Handler #4 >>BF86  
 DFFF \*\*\*\*\* SYSTEM ERROR HANDLER \*\*\*\*\*  
 DFFF Save Error Code (BF0F)  
 E003 Pop out of subroutine  
 E004 Exit to caller with Error Code (BF0F)  
 E008 RETURN  
 E009 \*\*\*\*\* SYSTEM DEATH HANDLER \*\*\*\*\*  
 E009 Save Error number in X-REG  
 E00A Turn off 80 column card (C00C)  
 E00D Select standard text display (C051)  
 E010 Are we running on a IIGS? (2278)  
 E013 No. >>E01A  
 E015 Yes, initialize IIGS video  
 E017 by clearing NEWVIDEO. (C029)  
 E01F ---  
 E021 Blank next to last row of screen and (0750)  
 E024 print "INSERT SYSTEM DISK AND RESTART" (FDE6)  
 E027 on bottom row of screen. (07D0)  
 E02D Get error number back  
 E02E Expect errors in range 00 to 0F  
 E030 Make it ASCII  
 E038 Put error number on screen (07F7)  
 E03B Infinite loop >>E03B  
 E03E \*\*\*\*\* PERFORM FILING OR \*\*\*\*\*  
 \*\*\*\*\* HOUSEKEEPING FUNCTIONS \*\*\*\*\*  
 E03E Save function index (FE7F)  
 E041 Get INFO flags for this command (FD95)  
 E044 Times 2  
 E045 Store Command Number times 2 (FE7B)  
 E04A And use it to index into Address Table  
 E04E Set up Jump Vector with this function's (FEBD)  
 E051 ..handler address (FD6E)  
 E057 Signal Backup required after call  
 E05C PATHNAME not required? >>E063  
 E05E Required - parse and validity check <E081>  
 E061 Bad Name? >>E07A  
 E063 Reference Number in list? (FE7B)  
 E066 No >>E06D  
 E068 Yes - check it out <E1C7>  
 E06B Bad Number? >>E07A  
 E06D Date/Time in list? (FE7B)

ProDOS MLI -- V1.2 -- 6 SEP 86      NEXT OBJECT ADDR: E070      NEXT OBJECT ADDR: E0F3

ProDOS MLI -- V1.2 -- 6 SEP 86      NEXT OBJECT ADDR: E070

ADDR    DESCRIPTION/CONTENTS

ADDR    DESCRIPTION/CONTENTS

```

E070 No >>E075
E072 Yes - set System date just in case <BF06>
E075 Call Function Handler <E07E>
E078 If no errors then exit >>E07D
E07A Else - call System error handler <BF09>
E07D Return to caller
E07E Indirect JUMP to Handler >>FEBD

E081 ***** CHECK CALLER'S PATHNAME *****
***** COPY TO MY AREA *****

E081 Set ($48) -> Pathname
E08C ---
E090 Assume partial Pathname (FE84)
E093 No Pathname in my area yet (D700)
E096 Check length of caller's Pathname
E098 Zero is no good >>E0F2
E09C Nor is 65 or more >>E0F2
E09E Save length (FE66)
E0A1 Length + 1 (FE66)
E0A5 Get first character of his name
E0A9 Is it "/"?
E0AB No >>E0B1
E0AD Yes - indicate fully qualified name (FE84)
E0B0 Bump past "/"
E0B1 ---
E0B3 Length of Index level is -1 initially (D700)
E0B6 First character of Index level (counter) (FE80)
E0B9 Start of upcoming Index level in name (FE82)
E0BC At end of name yet? (FE66)
E0BF Yes >>E0F6
E0C1 No - get next character in his name
E0C7 Is it "/"?
E0C9 Yes >>E10B
E0CB No - lower case?
E0CD No >>E0D1
E0CF Yes - force upper case
E0D1 Copy to my Pathname buffer (D700)
E0D4 Increment Index level counter (FE80)
E0D7 Subsequent characters may be A-Z,0-9 or . >>E0DE
E0D9 Increment Index level counter (FE80)
E0DC First character must be alphabetic >>E0EA
E0DE Is it "."?
E0E0 Yes - get next character >>E0BC
E0E2 No - is it special or control character
E0E4 Yes - Bad Pathname then >>E0F2
E0E6 Is it numeric?
E0E8 Yes - get next character >>E0BC
E0EA Is it Alphabetic?
E0F0 If so get next character >>E0BC
E0F2 Else

E0F3 Bad Pathname
E0F5 RETURN
E0F6 ---
E0F8 Any characters in last Index level? (FE80)
E0FB Yes >>E101
E0FD No, zero characters in it (FE80)
E100 And toss out last "/"
E101 ---
E102 Mark end of name with $00 (D700)
E105 Name too long? >>E0F2
E107 No - save final length (FE66)
E10A Set X -> 0
E10E Last Index more than 15 characters?
E110 Yes - then no good >>E0F2
E112 Save output Index (FE85)
E115 Store length of previous Index level (FE82)
E118 Just before it in buffer (D700)
E11B Restore output index (FE85)
E11E And continue >>E0B1
E120 End of Name
E121 Fully qualified name? (FE84)
E124 Yes >>E12B
E126 No - Got a Prefix (BF9A)
E129 No - error >>E0F2
E12B Else, okay to exit

E12C ***** MLI SET PREFIX CALL *****
***** MLI SET PREFIX CALL *****

E12C Copy Pathname <E081>
E12F It's okay >>E13B
E131 Check length of Volume name (D700)
E136 If zero - no Prefix wanted (BF9A)
E139 Exit with no error
E13A RETURN

E13B Get File entry for last index <E593>
E13E Okay? >>E144
E140 Invalid Pathname?
E142 No - Out now! >>E182
E144 Sub Directory file? (FE27)
E14B No, error >>E180
E14D Fully qualified path? (FE84)
E150 Yes >>E155
E152 No - use old Prefix also (BF9A)
E155 ---
E157 Compute new Prefix Index (FE66)
E15A Does new Prefix exceed 64 characters?
E15C Yes - Bad Path error >>E0F2
E15F Store new Prefix pointer (BF9A)

```

PRODOS MLI -- V1.2 -- 6 SEP 86 NEXT OBJECT ADDR: E165

PRODOS MLI -- V1.2 -- 6 SEP 86 NEXT OBJECT ADDR: E1FE

ADDR DESCRIPTION/CONTENTS

ADDR DESCRIPTION/CONTENTS

E165 Set Device Number of Prefix Directory (FE67)  
 E16B Save Keyboard for Prefix Directory (FE68)  
 E174 Copy Prefix to top of Path buffer (D7000)  
 E177 (preceded by old Prefix if one exists) (D7000)  
 E17F Exit normally  
 E180 Bad File Type Error  
 E182 ---  
 E183 RETURN

E1FE (\$48) -> 2nd Block of Buffer (index)  
 E200 ---  
 E201 Search all Volume Control Blocks (D910)  
 E204 for the one which goes with requested unit (D801)  
 E209 ---  
 E20F Can't find matching Volume Control Block  
 E211 So die with error type \$0A <BF0C>  
 E214 No Buffer in Open File Control Block  
 E216 So die with error type \$0B <BF0C>  
 E219 Is Volume mounted? (D9000)  
 E21C No, keep looking >>E209  
 E21E Save Volume Control Block index (FE59)  
 E222 Exit normally

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

E184 Set (\$4E) -> Data Buffer  
 E190 Set Length = 64 (max)  
 E19A Validity check buffer storage <FC46>  
 E19D Error? >>E182  
 E1A1 Get Prefix index (BF9A)  
 E1A5 No Prefix? - Length = 0 >>E1AB  
 E1A7 Complement for length  
 E1AB Store in first byte of buffer  
 E1AD If null Prefix exit >>E1C5  
 E1AF ---  
 E1B0 Copy Prefix to caller's buffer replacing (D7000)  
 E1B3 index level name length bytes with "/"  
 E1BD ---  
 E1C1 End it with a "/"  
 E1C5 ---  
 E1C6 Exit normally

E1C7 \*\*\*\*\* VALIDIFY CHECK REFERENCE NUMBER \*\*\*\*\*  
 (PASSED BY CALLER)

E1C7 Get Reference Number  
 E1CB If zero then no good >>E228  
 E1CF If > 8 then no good >>E228  
 E1D1 Save Reference Number  
 E1D2 Multiply by 32  
 E1D8 Result gives offset into PCB's (FE5A)  
 E1DC Get back Reference Number  
 E1DD File Control Block active this Reference? (D8000)  
 E1E0 No - Bad Reference Number >>E223  
 E1E2 Get Buffer Number (D80B)  
 E1E5 Find Buffer address in Global Page <FC000>  
 E1EB No Buffer? >>E214  
 E1ED Buffer okay, save Page Pointer in \$48  
 E1F1 Second block in \$49  
 E1F3 Set last Device used in Global Page (D801)  
 E1F9 Finish setting up pointers (FEA5)  
 E1FC (\$4A) -> 1st Block of Buffer (data)

E22C \*\*\*\*\* MLI ONLINE CALL \*\*\*\*\*  
 \*\*\*\*\* 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? <FC46>  
 E24C No, then exit >>E281  
 E24E Yes, zero out Buffer  
 E253 ---  
 E258 Index into Data Buffer = \$00 (FE32)  
 E25D Get Unit Number again  
 E25F Isolate valid bits  
 E261 Specific Unit requested? >>E282  
 E263 No, copy Device List from Global Page <E847>  
 E266 Save Device Count (FE85)  
 E269 Get last Device (FE92)  
 E26C Generate return data for it <E282>  
 E26F Bump data buffer index by 16 (FE82)  
 E278 Get next Device (FE85)  
 E27C And go do it >>E266  
 E27E When done, exit  
 E281 RETURN

ProDOS MLI -- V1.2 -- 6 SEP 86      NEXT OBJECT ADDR: E281  
 ADDR    DESCRIPTION/CONTENTS

E282 Save Device Number (BF30)  
 E285 Scan for the Volume Control Block <E859>  
 E288 Error? >>E2C5  
 E28A No  
 E28E Read block 2 (Volume Directory) <E8C9>  
 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 >>E2AF  
 E2AA Yes, Files open? (D911)  
 E2AD Yes >>E2BB  
 E2AF No, set up Volume Control Block for new VOL <E8B4>  
 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 >>E2E3  
 E2C5 \*\*\*\*\* ERROR \*\*\*\*\*  
     Store code in data buffer entry

---  
 E2C5 Store Device Number in entry <E2F8>  
 E2C6 Store error code next  
 E2CD Duplicate Volume error?  
 E2CF No - done >>E2E1  
 E2D2 Store Device Number for duplicate next (FE7E)  
 E2DA No Duplicate now  
 E2E1 Exit with error  
 E2E2 RETURN

E2E3 \*\*\*\*\* MAKE ONLINE VOLUME ENTRY \*\*\*\*\*  
 E2E3 Get name length for loop index (D900)  
 E2EC 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

ProDOS MLI -- V1.2 -- 6 SEP 86      NEXT OBJECT ADDR: E303  
 ADDR    DESCRIPTION/CONTENTS

E304 \*\*\*\*\*  
     \*\*\*\*\* MLI CREATE CALL \*\*\*\*\*  
     \*\*\*\*\*  
 E304 Follow Path to File <E5A6>  
 E307 Error? - I'm expecting one >>E30D  
 E309 If File was found - Duplicate error  
 E30B ---  
 E30C Return to caller  
 E30D File not found?  
 E30F NO, then a real error occurred >>E30B  
 E311 Yes, get requested storage type  
 E315 Is it 00, \$01, \$02 or \$03?  
 E317 Yes, carry on >>E31D  
 E319 Is it \$0D?  
 E31B No, then exit with error >>E32D  
 E31D Get status of this device (BF30)  
 E323 Exit on error >>E330  
 E325 Is there a free Directory entry? (FE63)  
 E328 NO >>E331  
 E32A Yes - continue >>E3BF  
 E32D Indicate Bad Storage Type  
 E330 Return to caller  
 E331 Is this the Volume Directory? (FE0E)  
 E337 NO, we can extnd it >>E33D  
 E339 Yes, indicate Volume Directory Full error  
 E33C Return to caller

## \* EXTEND DIRECTORY FILE \*

E33D Save old current Block number  
 E343 Allocate a Block on Disk <EA9C>  
 E346 Replace the number  
 E347 Replace BLKNUM  
 E34D Was there a free Block?  
 E34E No, then exit >>E330  
 E350 Yes, set up forward pointer in old one (DC02)  
 E353 to point to it (DC03)  
 E356 and Write old Directory Block <EBD5>  
 E359 Error? Yes, then exit >>E330  
 E35D Set BLKNUM -> new Block number  
 E362 Back point to old Directory Block (DC02)  
 E368 Loop until done >>E35D  
 E36C Zero remainder of Block Buffer (DC02)  
 E36F (including forward pointer) (DD00)  
 E373 Loop until done >>E36C  
 E375 Write new Directory Block <EBD5>

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

NEXT OBJECT ADDR: E378

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

NEXT OBJECT ADDR: E43C

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

E378 Error? Yes, then exit >>E330  
 E37A Set BLKNUM -> Parent Directory block number (FE0E)  
 E380 Read Block with my entry <EBC9>  
 E383 Entry number within the Parent Dir. block (FE10)  
 E386 None relocatable!!  
 E388 Set (\$48) -> Buffer  
 E38A Skip link pointers  
 E38C ---  
 E38D Count entries  
 E390 Skip to next (FE11)  
 E399 Save LSB  
 E39F Add 1 to Blocks used  
 E3A2 in entry  
 E3A8 Loop until done >>E39D  
 E3AA Write back Block to Parent Directory <EBD5>  
 E3AD Error? then exit >>E3BE  
 E3AF Start all over now that there's room >>E304

E3B2 \*\*\*\*\* ZERO \$F600 \*\*\*\*\*

E3B2 Zero \$F600 Block Buffer  
 E3BE Return to caller

E3BF \*\*\*\*\* BUILD NEW FILE \*\*\*\*\*

E3BF Call Zero \$F600 routine <E3B2>  
 E3C2 Copy Datetime (Creation)  
 E3C4 to my variables  
 E3D0 Loop until done >>E3C4  
 E3D2 Did he give Datetime (Creation)?  
 E3D3 Yes, carry on >>E3E0  
 E3D5 No, then use  
 E3D7 System Datetime instead (BF90)  
 E3E0 If Storage type is \$00, \$01, \$02 or \$03  
 E3E2 force it to \$10  
 E3E8 else use a \$D0  
 E3EA Find File name (FE82)  
 E3ED OR Storage type to name length (D700)  
 E3F0 Store Type/Length (FE27)  
 E3F3 Isolate name length  
 E3F7 Copy File name to File Entry Buffer (FE82)  
 E405 Copy caller's Access Byte  
 NOTE: This should be validity checked!!!  
 and copy File type  
 E40D  
 E412 ---  
 E413 and AUX\_TYPE  
 E41C Copy Version and Min Version (0,0) (FDB8)  
 E41F constants to entry (FE43)  
 E428 Indicate 1 Block used  
 E42D Copy Directory Header Block number (FE22)

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

E43C Is this a Seedling file?  
 E43E Yes >>E475  
 E440 No, Directory file - Build Header in \$F600  
 E442 Copy completed Directory entry (FE27)  
 E445 to \$F600 buffer first (DC04)  
 E449 Loop until done >>E442  
 E44B Make Storage type \$E in Header itself  
 E450 Put "HUSTON" (Author) in Reserved area  
 E458 and Version, Min Version, Access, (FDB8)  
 E45B Entry-length, File count and (DC20)  
 E45E Parent pointer from constants  
 E45F Loop until done >>E452  
 E463 EOF = \$200 (FE3D)  
 E466 Copy Parent Block entry number (FE24)  
 E46D Loop until done >>E466  
 E46F Copy Parent entry Length (FE19)  
 E475 Allocate a new disk block <EA9C>  
 E478 error? >>E4B1  
 E47A Store it in key pointer of entry (FE38)  
 E480 and in BLKNUM for I/O  
 E484 Write zeroed (or DIR HDR) key block <EBD5>  
 E487 error? >>E4B1  
 E489 Bump parent's file count (FE1B)  
 E491 Go update directory <E4B2>  
 E494 error? >>E4B1  
 E496 Checkpoint Volume Bit Map and exit >>EB76

E499 \*\*\*\*\* POINT \$48/49 AT DIRECTORY ENTRY \*\*\*\*\*

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

E4B2 \*\*\*\*\* UPDATE DIRECTORY(S) \*\*\*\*\*

E4B2 System date available? (BF90)  
 E4B5 no, forget it >>E4C2  
 E4B9 yes, copy to last modified date field (BF90)  
 E4C2 turn on BUBIT (backup) if appropriate (FE45)  
 E4CB set DEVNUM of parent (FE21)  
 E4D1 and BLKNUM (FE24)  
 E4D7 reread DIR block containing entry <EBC9>  
 E4DA error? >>E4B1  
 E4DC Point to proper entry in buffer <E499>  
 E4E3 Copy constructed entry to buffer (FE27)



```

ProDOS MLI -- V1.2 -- 6 SEP 86          NEXT OBJECT ADDR: E4EF
-----
ADDR  DESCRIPTION/CONTENTS
-----
E4EE  Is this block the DIR HDR block?
E4F9  no, write this modified directory block <EBD5>
E4FC  error? >>E4B1
E504  and then read DIR HDR block <EBC9>
E507  error? >>E4B1
E509  in any case..
E50B  copy back update file count to HDR (FE1B)
E514  and ACCESS byte (with Backup) (FE18)
E51A  re-write the HDR block <EBD5>
E51D  error? >>E573
E51F  is this the VOL DIR? (DC04)
E526  yes, all done -- exit >>E591
E528  Is subdirectory, get PARENT_ENTRY, (DC29)
E52B  store in variable area (FE26)
E52E  Get PARENT_ENTRY_LENGTH, (DC2A)
E531  store in variable area (FE19)
E534  get parent block number (DC27)
E53A  read Parent Directory block <EBC9>
E53D  error? >>E573
E53F  find entry for this subdirectory <E499>
E542  system date available? (BF90)
E545  no >>E554
E547  yes,
E54B  copy system date/time to... (BF90)
E54E  modified date/time in entry
E554  write it back <EBD5>
E557  error? >>E573
E55B  BLKNUM = HDR block number
E564  same block we have now?
E568  yes, go back and date stamp >>E51F
E56A  no,
E56E  read HDR block <EBD9>
E571  and go back to date stamp parent DIR >>E51F
E573  error? then exit

E574  ***** NOT PRODOS VOLUME ERROR *****
E574  ---
E577  RETURN

E578  ***** IS THIS PRODOS VOLUME? *****
E578  Does previous block_ptr = 0? (DC00)
E586  no, not a PRODOS volume >>E574
E588  else, (DC04)
E58D  does VOL DIR's STORAGE TYPE = $E or $F?
E58F  no, error >>E574
E591  else, ok
E592  RETURN

ProDOS MLI -- V1.2 -- 6 SEP 86          NEXT OBJECT ADDR: E592
-----
ADDR  DESCRIPTION/CONTENTS
-----
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 --> $F604 (HDR)
E5B7  copy part of HDR to file entry
E5C1  File type = $F (Directory) (FDB0)
E5C4  BLOCK = 2 (FE27)
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 <E6CD>
E5E2  got it! >>E644
E5E4  not yet, how many entries expected? (FE60)
E5E7  less entry number I just searched (FE5F)
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

```

ProDOS MLI -- V1.2 -- 6 SEP 86                      NEXT OBJECT ADDR: E617  
 -----  
 ADDR    DESCRIPTION/CONTENTS  
 -----

\*\*\* NO MORE FILE ENTRIES \*\*\*  
 E618 free entry found in directory? (FE63)  
 E61B yes >>E638  
 E61D no, check pointers (DC02)  
 E620 is there another block after this one? >>E627  
 E625 no... >>E638  
 E627 yes, free entry will be.. (FE24)  
 E630 first in that block  
 E635 indicate free entry available (FE63)  
 E638 find next index name <E764>  
 E63B exiting with error  
 E63C no more indicies in path, file not found >>E641  
 E63E else, path not found  
 E640 RETURN  
 E641 file not found error  
 E643 RETURN

## \*\*\* FOUND FILE ENTRY \*\*\*

E644 advance to next subdir in path <E75D>  
 E647 end -- save entry no. and exit >>E6B5  
 E64B get type of entry  
 E64F subdir?  
 E651 no, bad path then >>E63B  
 E655 copy key block no...  
 E657 to BLKNUM  
 E65A and to current DIR block no (FE22)  
 E664 go read key block of subdirectory <EBD9>  
 E667 error? >>E68D  
 E66C new file count (FE60)  
 E675 check minimum version (DC21)  
 E678 too new? >>E68B  
 E680 count bits in reserved field of DIR hdr  
 E681 --- >>E684  
 E684 ---  
 E687 there must be 5 bits on (normally \$75)  
 E689 (there are) >>E68F  
 E68B or else, incompatible file format  
 E68D ---  
 E68E RETURN  
 E68F copy DIR HDR <E695>  
 E692 and go scan for next level >>E5D4

ProDOS MLI -- V1.2 -- 6 SEP 86                      NEXT OBJECT ADDR: E692  
 -----  
 ADDR    DESCRIPTION/CONTENTS  
 -----

E695 \*\*\*\*\* COPY DIRECTORY HDR \*\*\*\*\*  
 E695 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  
 E6AB else, copy PARENT\_POINTER, (DC27)  
 E6AE PARENT\_ENTRY\_NO., and PARENT\_ENTRY\_LEN (FE0E)  
 E6B4 RETURN  
 E6B5 \*\*\*\*\* SAVE DIR ENTRY NO. & BLOCK \*\*\*\*\*  
 E6B5 compute entry number (FE1A)  
 E6BE save it (FE26)  
 E6C3 and the block it's in (FE24)  
 E6CC exit

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? (FE63)  
 E6E6 no >>E710  
 E6E8 yes, remember it <E6B5>  
 E6EB don't need another one now (FE63)  
 E6EE skip to next entry >>E710  
 E6F0 get length of name  
 E6F2 count it (FE5F)  
 E6F5 save it for loop (FE80)  
 E6FB same len as we are wanting? (D700)  
 E6FE no, skip it >>E710  
 E700 ---  
 E704 compare names (D700)  
 E70E we found it! exit  
 E70F RETURN  
 E710 skip to next entry (FE62)  
 E714 end of block? if so, exit >>E70F  
 E71A bump \$48/\$49 by entry len  
 E721 and go check next >>E6D9

ProDOS MLI -- V1.2 -- 6 SEP 86 NEXT OBJECT ADDR: E721

ADDR DESCRIPTION/CONTENTS

```

E723 ***** GET DIRECTORY DATA *****
E724 find base directory <E77C>
E725 error? >>E77B
E726 set out my variables (FE0E)
E727 zero up device number (BF30)
E728 copy DIR HDR to my variables <E695>
E729 copy TOTAL BLOCKS from VCB (D912)
E730 copy BIT MAP Pointer from VCB (D91A)
E731 copy Block No. of this directory (0046)
E732 make second copy of file count (FE1B)
E733 advance to next subdir in path <E764>
E734 and update index (FE82)
E735 RETURN

E764 ***** ADVANCE TO NEXT DIR NAME *****
E765 get this DIR's index (FE82)
E766 add len of name to move index to next name (FE82)
E767 still in prefix portion? >>E777
E768 no, now starting caller's path suffix (BF30)
E769 save last DEVNUM accessed (FE67)
E770 return with len of next dir in path (D700)
E771 RETURN

E77C ***** FIND BASE DIRECTORY *****
E77D get old PFXPTR (BF9A)
E77E fully qualified pathname? (FE84)
E77F no >>E787
E780 yes, no old PFXPTR anymore
E781 save old prefix index (FE83)
E782 DEVNUM=0 (BF30)
E783 ---
E784 *** SCAN VCB'S FOR A MOUNTED VOLUME ***
E785 scan (D900)
E786 got one >>E79F
E787 else, bump to next VCB
E788 ---
E789 *** FIND LAST DIR IN PREFIX OR TOL DIR ***
E790 store name length (FE80)
E791 same name as in pathname? (D700)
E792 no -- skip it >>E794
E793 save VCB index (FE59)
E794 DEVNUM = VCB's unit no. (D910)
E795 BLOCK = 2 (read VOLDIR if no old PFX)
E796

```

ProDOS MLI -- V1.2 -- 6 SEP 86 NEXT OBJECT ADDR: E7C4

ADDR DESCRIPTION/CONTENTS

```

E7C4 get old prefix index (FE83)
E7C5 ---
E7C6 accumulate a new index (FE82)
E7C7 no previous prefix? >>E7DD
E7C8 find last name in prefix (D700)
E7C9 read prefix directory instead of vol dir (FE68)
E7DA read block <EBD9>
E7DB error? >>E7E7
E7DC is this the right directory? <E881>
E7DD Yes---exit. >>E80B
E7DE *** IF NOT THERE, REMOUNT ALL VOLS ***
E7DF *** AND CHECK THEM ***
E7E0 open files? (FE59)
E7E1 Yes, give up now >>E808
E7E2 else, (FE83)
E7E3 put back old prefix length (FE82)
E7E4 copy DVCLST from global page <E847>
E7E5 use last device accessed first >>E80C
E7E6 if none, get last in my device table (BF31)
E7E7 volume not found error
E7E8 RETURN

E80C ---
E80D search for device in device table (FE92)
E80E device not found >>E808
E80F when found, make it active device (BF30)
E810 remove it from table (FE92)
E811 find its VCB <E859>
E812 not found? >>E846
E813 volume mounted there? (FE59)
E814 no >>E833
E815 yes, open files here? (D911)
E816 yes, skip it -- get next unit >>E7FD
E817 else read block 2 (vol dir)
E818 read volume directory <EBC9>
E819 error? >>E7FD
E820 mount volume on VCB <E8A7>
E821 error? >>E7FD
E822 is this his chosen volume? <E881>
E823 no, try again >>E7FD
E824 yes, exit
E825
E826 ***** COPY GLOB DEVLST TO MY TABLE *****
E827 start with last device (BF31)
E828 get a unit number (BF32)
E829 copy it to device table (FE92)
E830 return count of devices (BF31)
E831 RETURN

```

ProDOS MLI -- V1.2 -- 6 SEP 86      NEXT OBJECT ADDR: E858  
 ADDR    DESCRIPTION/CONTENTS

E859 \*\*\*\*\* SCAN VCB'S FOR DEVICE NO. \*\*\*\*\*

```

E859 ---
E85D scan VCB's for a given device number
E864 no it? >>E86B
E866 is it, save VCB index (FE59)
E869 and exit normally
E86A RETURN

E86B else, volume mounted here? (D900)
E86E yes >>E874
E871 no, save VCB index to empty unit (FE59)
E874 ---
E876 bump to next VCB
E878 and go look at it >>E85D
E87A not found...
E87B any free entries? if not, error >>E87E
E87D else, all is well -- return empty VCB
E87E VCB table full error
E880 RETURN
  
```

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

```

E881 ---
E886 check DIR type (DC04)
E889 VOL DIR or SUB DIR?
E88B neither >>E894
E88D yes..
E88F store len of its name (FE30)
E892 and go on >>E899
E894 error exit
E895 RETURN

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

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

```

E8A7 volume mounted? (FE59)
E8AD no, continue >>E8B4
E8AF yes, same one as one wanted? <E90C>
E8B2 if so exit, else fall thru >>E90B
  
```

ProDOS MLI -- V1.2 -- 6 SEP 86      NEXT OBJECT ADDR: E8B4  
 ADDR    DESCRIPTION/CONTENTS

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

```

E8B4 zero out VCB
E8BF is this a ProDOS volume? <E578>
E8C2 no -- exit >>E90B
E8C4 duplicate vol in VCB's? <E930>
E8C7 yes -- exit with that one instead >>E90A
E8C9 get new volume's name length (DC04)
E8D0 add to VCB index (FE59)
E8D4 and copy to VCB name field in empty VCB (DC04)
E8DF store in VCB name len field (D900)
E8E2 copy DEVNUM to VCB unit field (BF30)
E8E8 copy total blocks to VCB (DC29)
E8F4 copy block no. of vol dir to VCB
E8FE copy bit map block no. to VCB (DC27)
E90A exit
E90B RETURN
  
```

E90C \*\*\*\*\* COMPARE VOL NAMES TO MAKE \*\*\*\*\*  
 \*\*\*\*\* SURE THEY MATCH \*\*\*\*\*

```

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

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

```

E930 start with first VCB
E932 ---
E933 this VCB has same name? <E90C>
E936 no >>E947
E938 yes, files open? (D911)
E93B yes >>E951
E93F no, mark VCB empty (NAME=0) (D900)
E942 (UNIT=0) (D910)
E945 and exit with no error >>E94F
E947 else,
E949 bump to next VCB
E94D and loop >>E932
E94F exit no errors
E950 RETURN
  
```

PRODOS MLI -- V1.2 -- 6 SEP 86      NEXT OBJECT ADDR: E950  
 -----  
 ADDR    DESCRIPTION/CONTENTS  
 -----

E951 save flag (FE7D)  
 E954 and VCB index of duplicate vol (FE7E)  
 E957 exit with error  
 E958 RETURN

E959 \*\*\*\*\* SEE IF A QUANTITY OF FREE \*\*\*\*\*  
 \*\*\*\*\* BLOCKS IS AVAILABLE ON VOL \*\*\*\*\*

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

\*\*\* COMPUTE VCB FREE BLOCK COUNT \*\*\*

E964 no, how many bit map blocks are there? <EA08>  
 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 to buffer <EBD9>  
 E98C error? >>E9CA  
 E98E 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)  
 E9A5 no -- volume full >>E9C7  
 E9A7 save VCB bitmap block offset (D91C)  
 E9AA save free block count in VCB also (FE4F)  
 E9B6 are there enough to satisfy request? (D914)  
 E9C5 yes, exit  
 E9C6 RETURN

E9C7 volume full error  
 E9CA RETURN

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

E9CB scan through both buffer pages  
 E9D2 counting one bits <E9F8>  
 E9DD ---  
 E9E0 found free block already? (FE63)  
 E9E3 if so -- done >>E9F7  
 E9E5 any blocks found yet? (FE4E)  
 E9EB no >>E9F7  
 E9ED yes, compute total no. of bitmap blocks <EA08>  
 E9F1 less number remaining (FE64)

PRODOS MLI -- V1.2 -- 6 SEP 86      NEXT OBJECT ADDR: E9F4  
 -----  
 ADDR    DESCRIPTION/CONTENTS  
 -----

E9F4 gives bitmap block with first free bit (FE63)  
 E9F7 exit

E9F8 \*\*\*\*\* COUNT ONE BITS IN A BYTE \*\*\*\*\*

E9F8 shift and...

E9FB count bits that are on (FE4E)  
 EA03 exit when byte goes to zero  
 EA07 RETURN

EA08 \*\*\*\*\* COMPUTE NO. BITMAP BLKS -1 \*\*\*\*\*

EA08 get blocks on vol count (-1) (FE59)

EA14 ---  
 EA15 isolate top nibble of block count  
 EA16 for bit map block count  
 EA19 RETURN

EAL1 \*\*\*\*\* FREE A BLOCK ON DISK \*\*\*\*\*

EAL1 save MSB (FE64)  
 EALD and LSB  
 EA21 block number passed too big for (D913)  
 EA24 volume size? (FE64)  
 EA28 yes, error >>EA98  
 EA2B no, get bit position for block no.  
 EA31 save it (FE63)  
 EA35 divide block no. by 8 (FE64)  
 EA38 giving byte offset as remainder  
 EA41 save byte offset (FE6A)  
 EA44 make quotient/2 into block index (FE64)  
 EA47 remember which page in that block (FE6C)  
 EA4A read bit map block (after checkpoint) <EB43>  
 EA4D error? >>EA97  
 EA4F are we at proper block of bitmap yet? (FE71)  
 EA55 yes! >>EA6D  
 EA57 no -- checkpoint <EB76>  
 EA5A error? >>EA97  
 EA5C indicate block wanted in VCB (FE64)  
 EA65 DEVNUM of bitmap (FE6E)  
 EA68 read actual block directly <EB87>  
 EA6B error? >>EA97

EA6D get byte offset into page (FE6A)  
 EA70 which page? (FE6C)  
 EA73 get bit pattern to set (FE63)  
 EA76 page 0? >>EA80  
 EA78 no, turn bit on in page 1 (DB00)  
 EA7E and continue >>EA86  
 EA80 turn bit on in page 0 (DA00)  
 EA86 mark bitmap needs checkpoint  
 EA8E count block freed (FE8A)

```

ProDOS MLI -- V1.2 -- 6 SEP 86          NEXT OBJECT ADDR: EA96
-----
ADDR  DESCRIPTION/CONTENTS
-----

```

```

EA96 exit normally
EA97 RETURN

EA98 bad bitmap error
EA9B RETURN

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

EA9C go read bitmap <EB43>
EA9F error? >>EAC4
EAAB first page 0
EAAE scan 1st page of bitmap for free block(s) (DA00)
EAAE bump tm page 1 of buffer (FE6C)
EAB1 bump page offset (FE6B)
EAB4 scan 2nd page too (DB00)
EABC bump page (FE6B)
EABF get next block <EB28>
EAC2 continue >>EAAL
EAC4 error exit

EAC5 save byte index (FE6A)
EAC8 shift combination of page no. and (FE6B)
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)
EAE7 or page 1 (DA00)
EAEA shift bit pattern, bumping block no. LSB
EAEB until a one bit is found >>EAF0
EAF0 then shift it back the way it was
EAF1 (with that bit turned off) >>EAF0
EAF3 store LSB of block no. (FE4E)
EAF6 store updated byte.back in proper page (FE6C)
EB03 indicate bitmap needs checkpoint
EB0B one less block available in VCB (FE59)
EB20 ---
EB21 return with new block no. (FE4E)
EB27 RETURN

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

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

```

```

ProDOS MLI -- V1.2 -- 6 SEP 86          NEXT OBJECT ADDR: EB40
-----
ADDR  DESCRIPTION/CONTENTS
-----

```

```

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

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

EB72 disk full error
EB75 RETURN

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

EB76 ---
EB77 needs checkpoint? (FE6D)
EB7A no >>EB71
EB7C yes, write it <EBD1>
EB7F error? >>EB71
EB81 doesn't need checkpoint now
EB86 exit

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

EB87 save DEVNUM (FE6E)
EB8A copy block offset wanted (FE59)
EB94 BITMAP BLOCK = BITMAP PTR + BLOCK OFFSET (D91A)
EBA2 set up read command

*** READ OR WRITE BITMAP ***

EBA4 save I/O command
EBA8 device = bitmap device (FE6E)
EBB0 block = bitmap block (FE6F)
EBBA point to bitmap buffer (EA82)
EBBD do the I/O <EBDF>
EBC2 restore old DEVNUM (BF30)
EBC5 ok? >>EBC8
EBC7 no, error exit
EBC8 RETURN

```

PRODOS MLI -- V1.2 -- 6 SEP 86      NEXT OBJECT ADDR: EBC8  
 -----  
 ADDR    DESCRIPTION/CONTENTS  
 -----

EBC9 \*\*\*\*\* READ BLOCK DESIGNATED BY A,X \*\*\*\*\*  
 EBCB Put low byte of block number in BLKNUM  
 EBCD and high byte in BLKNUM+1  
 EBD0 Read a block <EBD9>  
 EBD0 RETURN

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

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

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

EBDB \*\*\*\*\* READ OR WRITE BLOCK \*\*\*\*\*  
 EBD8 save I/O command  
 EBD9 where is my buffer?  
 EBD0 save flags  
 EBE0 and disable  
 EBE3 Set low byte of Buffer pointer  
 EBE5 to zero  
 EBE7 Initialize Global Page System error to 0 (BF0F)  
 EBEA set I/O transfer occurred flag  
 EBEF set unit to do I/O on (BF30)  
 EBF4 do block I/O <DEF4>  
 EBF7 error? >>EBFC  
 EBF9 no errors, restore things and exit  
 EBF8 RETURN

EBFC error exit  
 EBFE RETURN

EBFF \*\*\*\*\* MLI GET MARK CALL \*\*\*\*\*  
 \*\*\*\*\* MLI GET MARK CALL \*\*\*\*\*

EBFF copy mark to caller's list from FCB (FE5A)  
 EC0F exit with no errors  
 EC10 RETURN

PRODOS MLI -- V1.2 -- 6 SEP 86      NEXT OBJECT ADDR: EC10  
 -----  
 ADDR    DESCRIPTION/CONTENTS  
 -----

EC11 bad position error  
 EC14 RETURN

EC15 \*\*\*\*\* MLI SET MARK CALL \*\*\*\*\*  
 \*\*\*\*\* MLI SET MARK CALL \*\*\*\*\*

EC15 set up to...  
 EC1D copy user's mark to temporary  
 EC1F new mark variable (FE70)  
 EC24 make sure it will not exceed EOF (D815)  
 EC29 else, error >>EC11  
 EC2C ---

\*\*\* STILL IN SAME DATA BLOCK? \*\*\*

EC32 get old mark (FE5A)  
 EC35 find its block no. (\*2) (D813)  
 EC3D compute distance in pages from old mark's (FE73)  
 EC41 block to new mark (FE4E)  
 EC47 earlier -- need new data block >>EC58  
 EC4B too far forward -- need new block >>EC58  
 EC50 MSB's match? (D814)  
 EC55 then mark is still in this block >>ED79

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

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

\*\*\* NEED DIFFERENT DATA BLOCK \*\*\*

EC6D copy storage type (D807)  
 EC73 old data block needs writing? (D808)  
 EC78 no >>EC7F  
 EC7A yes, do so <EE83>  
 EC7D error? >>ECE8  
 EC7F see if new mark is outside the range of (FE5A)  
 EC82 the current index block (D814)  
 EC91 yes >>ECB1  
 EC95 yes >>ECB1  
 EC97 no, same index block (FE5E)  
 EC9A check storage type

```

ProDOS MLI -- V1.2 -- 6 SEP 86      NEXT OBJECT ADDR: EC9B
-----
ADDR  DESCRIPTION/CONTENTS
-----

```

```

EC9B  sapling or tree are ok >>ED16
      *** SEEDLING ***
EC9D  seedling, check position (FE73)
ECA0  if position is outside of block 0..
ECA4  promote to sapling >>ED04
ECA6  else, (D80C)
ECAE  go get key block (seedling data block) >>ED6F

```

```

      *** NEED TO CHANGE DATA BLOCKS ***
ECB1  does old index block need dumping? (D808)
ECB6  no >>ECBD
ECB8  yes, do so <EE97>
ECBB  error? >>ECE8
ECBD  check storage type (FE5E)
ECC0  tree file?
ECC2  yes >>ECE9
ECC4  no, sapling (FE74)
ECC9  is position in first index block?
ECCC  no, need master index, subindex and data >>ED2F
ECCE  yes, first index, reset flags <ED9F>
ECD1  is this a seedling?
ECD2  if so, see if in first block >>EC9D

```

```

      *** SAPLING ***
ECD4  no, sapling, read its only index block <EE2A>
ECD7  error? >>ECE8
ECDC  set block no. of index block
ECE6  Always branch >>ED16

```

```

ECE8  Error exit

```

```

      *** TREE FILE/NEED ANOTHER INDEX BLOCK ***
ECE9  reset flags <ED9F>
ECEC  read master index block <EE2A>
ECEF  error? >>ECE8
ECF4  MSB_of_position/2
ECFA  is there a subindex there?
ECFC  yes! >>ED09
ED02  no, fall thru to make one
      *** GET NEW INDEX BLOCK ***

```

```

ProDOS MLI -- V1.2 -- 6 SEP 86      NEXT OBJECT ADDR: ED04
-----
ADDR  DESCRIPTION/CONTENTS
-----

```

```

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? >>ECE8

```

```

      *** SAPLING/TREE - THIS INDEX BLOCK ***
ED16  make block no. out of position (FE74)
ED1F  use as an index to examine index block
ED21  entry
ED27  if its zero....
ED2B  need new data block
ED2F  set flags for what to allocate (FE5A)
ED38  new index block being created?
ED3A  zero data block in any case <ED57>
ED3D  if not index block that's it >>ED79
ED3F  Zero the Index Block I/O Buffer <ED45>
ED42  and continue >>ED79

```

```

ED45  ***** ZERO INDEX BLOCK I/O BUFFER *****
ED45  ---
ED48  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 from index entry
ED6F  ---
ED71  read new data block <EDF4>
ED74  error? >>ED9E
ED76  reset block allocation flags <ED9F>

```



ProDOS MLI -- V1.2 -- 6 SEP 86      NEXT OBJECT ADDR: ED76  
 ADDR    DESCRIPTION/CONTENTS

ProDOS MLI -- V1.2 -- 6 SEP 86      NEXT OBJECT ADDR: EDEF  
 ADDR    DESCRIPTION/CONTENTS

\*\*\* GOT DATA BLOCK WANTED \*\*\*  
 ED79 ---  
 ED80 save previous mark in my variables (D812)  
 ED86 set new mark in the FCB (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

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

ED9F \*\*\*\*\* RESET BLOCK ALLOC FLAGS \*\*\*\*\*  
 ED9F get flags (FE5A)  
 EDA5 turn off low 3 bits (allocate no new  
 EDA7 blocks to file) (D808)  
 EDAA RETURN

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

EDAB \*\*\*\*\* SET DIR FILE POSITION \*\*\*\*\*  
 EDAB DIR file?  
 EDAD yes! >>EDB4  
 EDAD no, bad storage type error  
 EDAF go to SYSERR <BF09>  
 EDB1 else, get page distance (FE4E)  
 EDB4 make it into blocks (divide by 2)  
 EDB7 new position beyond old? (FE73)  
 EDC1 yes >>EDD1  
 EDC3 else, use previous mark  
 EDC5 copy to BLKNUM <EDDF>  
 EDCB error? >>EDEC  
 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? >>EDEC  
 EDD8 count it (FE62)  
 EDDB more to skip >>EDD1  
 EDDD got it now! >>ED79

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 FCB (FE5A)  
 EE34 use \$48/\$49 buffer  
 \*\*\* I/O BLOCK \*\*\*  
 EE36 set I/O command  
 EE38 and block no. (D800)  
 EE42 must be non-zero block number  
 EE46 or horrible death!  
 EE4B fall through to read/write block (D801)

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

\*\*\* 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 FCB (D801)  
 EE63 set I/O transfer has occurred flag

PRODOS MLI -- V1.2 -- 6 SEP 86      NEXT OBJECT ADDR: EE68  
 ADDR    DESCRIPTION/CONTENTS

EE68 set unit no. from DEVNUM (BF30)  
 EE6D no errors have occurred yet  
 EE72 do block I/O <DEE4>  
 EE75 error? >>EE7A  
 EE77 no, exit normally  
 EE79 RETURN  
 EE7A else, exit with error  
 EE7C RETURN

EE7D \*\*\*\*\* CHECKPOINT BITMAP & KEY BLOCK \*\*\*\*\*  
 EE7D checkpoint bitmap buffer <EB76>  
 EE80 go write key block for file >>EE27

EE83 \*\*\*\*\* CHECKPOINT DATA BLOCK BUFFER \*\*\*\*\*  
 EE83 buffer pointer at \$4A/\$4B  
 EE85 point to block no. in FCB  
 EE8D go write buffer to disk <EE36>  
 EE90 error? >>EEB4  
 EE94 go turn off \$40 flag in FCB and exit >>EBAB

EE97 \*\*\*\*\* CHECKPOINT INDEX BLOCK BUFFER \*\*\*\*\*  
 EE97 checkpoint volume bitmap <EB76>

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

EEB5 \*\*\*\*\* MLI OPEN CALL \*\*\*\*\*  
 \*\*\*\*\*

EEB5 search path for file <E593>  
 EEB8 found it? >>EEBE  
 EEBA no, bad path error  
 EEEB exit >>EEC5  
 EEBE else, see if FCB already open on file <EF9B>  
 EEC1 for write. if not, continue. >>EECB  
 EEC3 else, file already open error  
 EEC5 ---  
 EEC6 RETURN

PRODOS MLI -- V1.2 -- 6 SEP 86      NEXT OBJECT ADDR: EEC6  
 ADDR    DESCRIPTION/CONTENTS

EEC7 Error -- unsupported storage type  
 EEEA RETURN

EECB get FCB index (FE5A)  
 EED1 free FCB found? >>EED7  
 EED3 no, all FCB's in use error  
 EED6 RETURN

EEED zero out unused FCB  
 EEE2 copy file ID fields to FCB  
 EEE5 (DEVNUM, DIR HDR BLK, DIR BLK, (FE5A)  
 EEE8 DIR ENTRY NO.)  
 EEEF isolate storage type (FE27)  
 EEFB and copy to FCB (D807)  
 EEFE get access (FE45)  
 EF03 DIR file?  
 EF05 no >>EF09  
 EF07 yes, we are only reading (I hope)  
 EF09 update access flag in FCB (D809)  
 EF0E write protected? >>EF15  
 EF10 no, another FCB open on this file? (FE5F)  
 EF13 yes, no touchie >>EEC3

EF15 storage type must be < \$4  
 EF19 or equal to \$D  
 EF1B else, storage type error >>EEC7  
 EF1D ---

EF1F copy key block, blocks used, and  
 EF21 EOF mark to FCB (FE5A)  
 EF31 BLKNUM = key block number  
 EF36 store REFNUM in FCB (FEG2)  
 EF3C go check and assign I/O buffer <FBB1>  
 EF3F error? >>EF65

EF41 go find VCB and set buff ptrs <E1E2>  
 EF44 set current level in FCB (BF94)  
 EF4A seedling, sapling or tree? (D807)  
 EF4F no, skip next stuff >>EF7C  
 EF51 yes, make current mark in FCB outside  
 EF53 first index block to force a read of all (D814)  
 EF56 index blocks and BLOCK 0.  
 EF5A zero mark wanted, however (FE72)  
 EF60 go set mark to zero <EC32>  
 EF63 ok? >>EF81  
 EF65 no, save the error code  
 EF69 got and I/O buffer? (D80B)  
 EF6C no >>EF74  
 EF6E yes, free it <FC0E>  
 EF74 mark FCB not in use  
 EF7A exit with error  
 EF7B RETURN

PRODOS MLI -- V1.2 -- 6 SEP 86 NEXT OBJECT ADDR: EF7B

-----  
 ADDR DESCRIPTION/CONTENTS

EF7C else, read key block to I/O buffer <EDF4>  
 EF7F error? >>EF65  
 EF81 bump open file count in VCB (FE59)  
 EF87 indicate files are open in VCB (D911)  
 EF8F put REF NUM in caller's parmlist (FE5A)  
 EF99 exit with no errors  
 EF9A RETURN

EF9B \*\*\*\*\* FIND A FCB \*\*\*\*\*

EF9B clear flags and index byte  
 ---  
 EFA7 found a free FCB yet? (FE5B)  
 EFAA yes >>EFAF  
 EFAC no, bump entry count (FE62)  
 EFAD FCB in use? (D800)  
 EFB2 yes >>EFC1  
 EFB4 no,  
 EFB7 save index to free FCB (FE5A)  
 EFB8 flag that we found one  
 EFBF and skip this FCB >>EFDF  
 EFC1 ---  
 EFC7 compare file ID's to see if this FCB (D800)  
 EFC8 is open on the requested file. (FE20)  
 EFCD no match? >>EFDF  
 EFD3 indicate FCB already open on file (FE5F)  
 EFD6 write enabled? (D809)  
 EFD8 if not, allow multiple open access to file >>EFDF  
 EFDD else, error exit  
 EFDE RETURN

EFDE return index to start of FCB  
 EFE3 bump to next FCB  
 EFE5 and loop >>EFA6  
 EFE7 when done, exit normally  
 EFE8 RETURN

EFE9 \*\*\*\*\*  
 \*\*\*\*\* MLI READ CALL \*\*\*\*\*  
 \*\*\*\*\*

EFE9 point to data buffer <FLF5>  
 EFEC copy request length <FLDA>  
 EFEF save access  
 EFFF set up marks <F207>  
 EFFF read access permitted?  
 EFFF yes >>EFFF  
 EFFF no, access error  
 EFFF will we read past EOF? >>F023  
 EFFF yes, (FE5A)

PRODOS MLI -- V1.2 -- 6 SEP 86 NEXT OBJECT ADDR: F001

-----  
 ADDR DESCRIPTION/CONTENTS

F001 LENGTH = EOF - current mark (D815)  
 F019 are we already at EOF? (FEA2)  
 F01C no >>F02E  
 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 <FC46>

F031 no good? >>F020  
 F033 ok, get storage type for file <F200>  
 F036 standard kind of file?  
 F038 yes >>F03D  
 F03A no, DIR file >>F1A3

F03D else, set mark (to read proper buffers) <EC32>  
 F040 error? >>F020  
 F042 set up buffer indexing <F0F8>  
 F045 move all that can be moved out of data buff <F122>  
 F048 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 >>F03D  
 F052 if so, store block count wanted (FE77)  
 F055 get FCB flags <F5D6>  
 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? >>F0D5  
 F06C bump buffer pointer to next location  
 F070 drop length remaining by 512 bytes (FE76)  
 F076 bump mark (FE73)  
 F07E and mark's MSB 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 ptrs back <F195>  
 F08F more to read? (FE75)  
 F095 no, exit through finish-up >>F0E1  
 F097 yes, conventional block by block read then >>F03D

PRODOS MLI -- V1.2 -- 6 SEP 86 NEXT OBJECT ADDR: F097  
 ADDR DESCRIPTION/CONTENTS

F099 crossed index block? go do set mark >>F067  
 F09B make index block offset from mark (FE74)  
 F0A4 BLKNUM = next block in index block  
 F0AA 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  
 F0C8 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 (D80A)  
 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

PRODOS MLI -- V1.2 -- 6 SEP 86 NEXT OBJECT ADDR: F121  
 ADDR DESCRIPTION/CONTENTS

F122 \*\*\*\*\* COPY FROM I/O BLOCK BUFF \*\*\*\*\*  
 \*\*\*\*\* TO DATA BUFFER \*\*\*\*\*  
 EXITS IF: LENGTH GOES TO ZERO  
 NEXT BLOCK IS NEEDED  
 NEWLINE IS FOUND  
 ON EXIT: OVERFLOW FLAG SET IF DONE  
 OVERFLOW ZERO IF NEXT BLOCK NEEDED

F122 ---  
 F123 partial page to move? >>F12D  
 F125 no, any full pages left? (FE76)  
 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 >>F150  
 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 mind! >>F134  
 F16F else, were we done with page?  
 F170 no >>F17C  
 F172 yes, bump pointer  
 F174 and mark (FE73)  
 F17C set overflow flag (read completed) (F194)  
 F17F update mark LSB (FE72)  
 F184 bump request count if necessary  
 F185 update count LSB (FE75)  
 F18B point beyond data in caller's buffer  
 F193 ---  
 F194 and exit

PRODOS MLI -- V1.2 -- 6 SEP 86 NEXT OBJECT ADDR: F194

PRODOS MLI -- V1.2 -- 6 SEP 86 NEXT OBJECT ADDR: F194

ADDR DESCRIPTION/CONTENTS

ADDR DESCRIPTION/CONTENTS

F195 \*\*\*\*\* CLEANUP AFTER DIRECT I/O \*\*\*\*\*  
 F195 restore caller's data buffer pointer  
 F1A0 go set buffers/find VCB and exit >>ELE2  
 F1A3 \*\*\*\*\* DIRECTORY FILE READ \*\*\*\*\*

F195 \*\*\*\*\* CLEANUP AFTER DIRECT I/O \*\*\*\*\*  
 F195 restore caller's data buffer pointer  
 F1A0 go set buffers/find VCB and exit >>ELE2  
 F1A3 \*\*\*\*\* DIRECTORY FILE READ \*\*\*\*\*

F1A3 set mark/read <EC32>  
 F1A6 error? >>F1D7  
 F1A8 set up buffer indexing <F0F8>  
 F1AB move data from I/O buffer <F122>  
 F1AE need next block? >>F1A3  
 F1B0 no, finish up I/O <F0E1>  
 F1B3 ok? exit >>F1D5  
 F1B5 not ok. EOF error?  
 F1B8 no, out now >>F1D6  
 F1BA yes, point beyond EOF anyway? <ED79>  
 F1BD zero out data block I/O buffer <ED57>  
 F1C5 dummy up an empty DIR block with previous (D810)  
 F1C8 pointer and no forward pointer in I/O  
 F1CA buffer.  
 F1CC zero out current block no. (D810)  
 F1D5 return to caller  
 F1D6 RETURN  
 F1D7 finish up and error exit >>F0DA

F1A3 set mark/read <EC32>  
 F1A6 error? >>F1D7  
 F1A8 set up buffer indexing <F0F8>  
 F1AB move data from I/O buffer <F122>  
 F1AE need next block? >>F1A3  
 F1B0 no, finish up I/O <F0E1>  
 F1B3 ok? exit >>F1D5  
 F1B5 not ok. EOF error?  
 F1B8 no, out now >>F1D6  
 F1BA yes, point beyond EOF anyway? <ED79>  
 F1BD zero out data block I/O buffer <ED57>  
 F1C5 dummy up an empty DIR block with previous (D810)  
 F1C8 pointer and no forward pointer in I/O  
 F1CA buffer.  
 F1CC zero out current block no. (D810)  
 F1D5 return to caller  
 F1D6 RETURN  
 F1D7 finish up and error exit >>F0DA

F1DA \*\*\*\*\* COPY CALLER'S I/O LENGTH \*\*\*\*\*  
 F1DA copy request length to LENGTH and  
 F1DC a temporary variable  
 F1DE pick up ACCESS flcgs for file (FE5A)  
 F1F3 exit to caller  
 F1F4 RETURN  
 F1F5 \*\*\*\*\* POINT \$4E/\$4F TO CALLER'S \*\*\*\*\*  
 \*\*\*\*\* DATA BUFFER \*\*\*\*\*  
 F1F5 set up pointer  
 F200 YREG --> FCB (FE5A)  
 F203 AREG = storage type (D807)  
 F206 exit  
 F207 \*\*\*\*\* COPY FILE MARK AND COMPUTE \*\*\*\*\*  
 \*\*\*\*\* AND COMPARE END MARK \*\*\*\*\*

F1DA \*\*\*\*\* COPY CALLER'S I/O LENGTH \*\*\*\*\*  
 F1DA copy request length to LENGTH and  
 F1DC a temporary variable  
 F1DE pick up ACCESS flcgs for file (FE5A)  
 F1F3 exit to caller  
 F1F4 RETURN  
 F1F5 \*\*\*\*\* POINT \$4E/\$4F TO CALLER'S \*\*\*\*\*  
 \*\*\*\*\* DATA BUFFER \*\*\*\*\*  
 F1F5 set up pointer  
 F200 YREG --> FCB (FE5A)  
 F203 AREG = storage type (D807)  
 F206 exit  
 F207 \*\*\*\*\* COPY FILE MARK AND COMPUTE \*\*\*\*\*  
 \*\*\*\*\* AND COMPARE END MARK \*\*\*\*\*

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 FCB (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 \*\*\*\*\*  
 \*\*\*\*\*

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 FCB (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 \*\*\*\*\*  
 \*\*\*\*\*

F26E copy request length <FIDA>  
 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 = 0? (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  
 F298 set mark/read blocks <EC32>  
 F29B error? >>F2C1  
 F29D get FCB flags <F5D6>  
 F2A0 any new blocks needed?  
 F2A2 no >>F306  
 F2A4 yes, allocating them  
 F2A6 ---

F26E copy request length <FIDA>  
 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 = 0? (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  
 F298 set mark/read blocks <EC32>  
 F29B error? >>F2C1  
 F29D get FCB flags <F5D6>  
 F2A0 any new blocks needed?  
 F2A2 no >>F306  
 F2A4 yes, allocating them  
 F2A6 ---

PRODOS MLI -- V1.2 -- 6 SEP 86 NEXT OBJECT ADDR: F207

PRODOS MLI -- V1.2 -- 6 SEP 86 NEXT OBJECT ADDR: F207

ADDR DESCRIPTION/CONTENTS

ADDR DESCRIPTION/CONTENTS

```

ProDOS MLI -- V1.2 -- 6 SEP 86          NEXT OBJECT ADDR: F2A7          NEXT OBJECT ADDR: F33A
-----
ADDR  DESCRIPTION/CONTENTS
-----

```

```

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 FCB flags <F5D6>
F2B8 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/EOF <F234>
F2C6 and finish I/O, exit with error >>F0DA

F2C9 check FCB 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 FCB flags <F5D6>
F2DD indicate index buffer changed
F2DF no new blocks needed now
F2E1 update FCB flags (D808)
F2E7 make index block offset from mark
F2EF store new block no. in index block (FE4F)
F2FC and store it as current data block (FE5A)
F306 set up buffer indexing <F0F8>
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 $100 (FE73)
F333 still in same I/O block page?
F337 yes >>F31C
F33A no, clear overflow (I/O incomplete) >>F361

```

```

ProDOS MLI -- V1.2 -- 6 SEP 86          NEXT OBJECT ADDR: F33A
-----
ADDR  DESCRIPTION/CONTENTS
-----

```

```

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 set FCB flag to indicate write occurred <FA2C>
F380 exit

```

```

F381 ***** ADD NEW MASTER INDEX BLOCK *****
      (MAKE A TREE FILE)

```

```

F381 add higher level <F3CA>
F384 error? >>F3C9
F386 get storage type <F200>
F389 tree?
F38B yes >>F392
F38D no, add another level <F3CA>
F390 error? >>F3C9
F392 buy another block <F411>
F395 error? >>F3C9
F397 male offset into current index block (FE74)
F39A from current mark
F39C point index to new block (FE4E)
F3AB also save as current data block (FE5A)
F3B5 checkpoint bitmap & key block <EE7D>
F3B8 error >>F3C9
F3BA zero out new index block >>ED45

```

```

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

```

```

F3BD check storage type <F200>
F3C2 seedling? >>F3CA
F3C4 no, read key index block <EE2A>
F3C7 and go add data block >>F392
F3C9 exit if error occurs

```

ProDOS MLI -- V1.2 -- 6 SEP 86                      NEXT OBJECT ADDR: F3C9  
 ADDR    DESCRIPTION/CONTENTS

ProDOS MLI -- V1.2 -- 6 SEP 86                      NEXT OBJECT ADDR: F45A  
 ADDR    DESCRIPTION/CONTENTS

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

F3CA buy a block <F411>  
 F3CD error? >>F410  
 F3D2 save old key block number (D80C)  
 F3DA make new block the key block (D80C)  
 F3E7 and current index block in FCB (D80F)  
 F3F0 store pointer to old key block  
 F3F3 in first position of new index  
 F3FA checkpoint bitmap and new key block <EE7D>  
 F3FD error? >>F410  
 F3FF get storage type <F200>  
 F404 upgrade it to next higher type (D807)  
 F407 indicate DIR entry needs update (D808),  
 F410 exit

F411 \*\*\*\*\* BUY A DISK BLOCK \*\*\*\*\*

F411 allocate a disk block <EA9C>  
 F414 error? >>F430  
 F416 get FCB flags <F5D6>  
 F419 indicate DIR entry needs update  
 F422 add 1 to blocks in use for file  
 F42F ---  
 F430 exit

F431 \*\*\*\*\* DO STATUS IF NO I/O YET \*\*\*\*\*

F431 get FCB flags <F5D6>  
 F434 any buffers in use? (I/O activity)  
 F436 if so, assume its ok >>F42F  
 F438 no, (D801)  
 F43B select new device (BF30)

\*\*\* STATUS CALL \*\*\*

F43E Save Unit Number  
 F440 Save Block Number on stack  
 F446 Indicate Status call  
 F44A Indicate Block 0  
 F44E Go do I/O <DEE4>  
 F451 Restore Block Number to original value  
 F459 Exit

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

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

F45A check REF NUM  
 F45E specific close? >>F494

\*\*\* CLOSE ALL OPEN FILES \*\*\*

F460 no errors yet (FE89)  
 F465 store FCB index (FE5D)  
 F469 get its level (D81B)  
 F46C if below system LEVEL, skip it (BF94)  
 F46F yes, skip it >>F486  
 F471 no, active FCB? (D800)  
 F474 no >>F486  
 F476 yes, flush it and update directory <F4F2>  
 F479 error? >>F4C7  
 F47B no, close specific FCB <F499>  
 F480 is this a close-all?  
 F482 yes, ignore errors >>F486  
 F484 no, stop on error >>F4C7  
 F486 bump FCB index to next one (FE5D)  
 F48C and continue >>F465  
 F48E when all FCBs checked, load error number (FE89)  
 F491 no error >>F4C5  
 F493 error exit

\*\*\* CLOSE SPECIFIC FILE \*\*\*

F494 flush it <F4FA>  
 F497 error? >>F4C7  
 F499 get buffer number (FE5D)  
 F49F free its pages <FC11>  
 F4A2 error? >>F4C7  
 F4A4 release FCB  
 F4AC set DEVNUM (D801)  
 F4B2 find VCB for device <E859>  
 F4B5 decrement count of open files in VCB (FE5C)  
 F4BB some are open... >>F4C5  
 F4BD if all are closed, turn off (D911)  
 F4C0 "files open" flag  
 F4C5 ---  
 F4C6 exit

F4C7 Branch to handle close error >>F4F7

PRODOS MLI -- V1.2 -- 6 SEP 86      NEXT OBJECT ADDR: F4C7

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

F4C9 \*\*\*\*\* MLI FLUSH CALL \*\*\*\*\*  
 \*\*\*\*\* MLI FLUSH CALL \*\*\*\*\*  
 \*\*\*\*\* MLI FLUSH CALL \*\*\*\*\*  
 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)  
 F4D8 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 ---  
 F4EB 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 <E1C7>  
 F502 error? >>F4F7  
 F504 is write access allowed? (D809)  
 F509 no, exit >>F4EA  
 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 >>F4EA  
 F517 else, get FCB flags <F5D6>  
 F51A has data buffer changed?  
 F51C no >>F523  
 F51E yes, checkpoint it <EB83>  
 F521 error? >>F4F7  
 F523 get flags again <F5D6>  
 F526 has index buffer changed?  
 F528 nm >>F52F  
 F52A yes, checkpoint it <EB97>  
 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 <EB09>

PRODOS MLI -- V1.2 -- 6 SEP 86      NEXT OBJECT ADDR: F54C

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

F54C error? >>F4F7  
 F54E copy directory header <E695>  
 F551 are we in block with this file's entry? (FE27)  
 F55A no >>F561  
 F55F yes >>F568  
 F561 no, set new block number  
 F565 read it <EBD9>  
 F568 point at directory entry in block <E499>  
 F56B copy file entry from directory <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)  
 F5A8 write entry back to directory <E4B2>  
 F5AB error? >>F5C7  
 F5B0 turn off "write occurred" flag (D81C)  
 F5B8 same bitmap in memory (FE24)  
 F5BE no, exit now >>F5C5  
 F5C0 yes, checkpoint it also <EB76>  
 F5C5 no errors, exit  
 F5C6 RETURN

F5C7 \*\*\*\*\* CLOSE ERROR \*\*\*\*\*

F5C7 is this a close or flush all?  
 F5CC no >>F5D4  
 F5D0 yes, save error code (FE89)  
 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 \*\*\*\*\* MLI SET EOF CALL \*\*\*\*\*  
 \*\*\*\*\* MLI SET EOF CALL \*\*\*\*\*



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

NEXT OBJECT ADDR: F5E1

ADDR DESCRIPTION/CONTENTS

```

F5E1 get storage type <F200>
F5E4 if DIR file...
F5E6 its an access error >>F5DD
F5E8 else, save type for truncate to
F5E9 mess with.
F5EF write access permitted? (D809)
F5F4 no, error >>F5DD
F5F6 check device status <F431>
F5F9 error? >>F5DD
F602 copy EOF from FCB (D815)
F610 copy caller's new EOF
F61B compare old EOF to new (FE55)
F621 if less than or equal to... >>F628
F623 if greater... >>F63D

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

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

*** OLD EOF > NEW EOF ***
*** 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 EOF (FE5D)
F659 it is prior to EOF >>F672
F661 if past EOF, force mark back to EOF (FE5D)
F672 construct EOF block number and (FE75)
F675 byte offset into block from new (FE91)
F678 EOF mark. (FE76)
F690 on a block boundary? (FE92)
F693 yes >>F6B2
F695 no, (FE90)
F699 decrement block by 1
F6A7 but don't let it fall below 0
F6B2 copy key block number (FE5D)
F6C1 set blocks freed to zero
F6C9 truncate file at new EOF <FA3E>
F6CC save status
F6DA set new key block in FCB (FE8A)
F6DA drop FCB block count by number (D818)
F6DD of blocks freed in truncate routine. (FE8D)
F6EA copy new storage type (FE8C)
F6F7 turn off all block allocation flags <ED9F>
F6FA update VCB free block count <F9BD>
F704 copy mark (D812)

```

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

NEXT OBJECT ADDR: F70C

ADDR DESCRIPTION/CONTENTS

```

F70C force current mark to infinity (D812)
F713 go set mark <EC32>
F716 no errors? >>F71F
F718 if error, indicate in saved status
F71E but continue
F71F copy caller's EOF to FCB <F628>
F722 Flush and update <F4FA>
F725 no errors? >>F72E
F727 if error, indicate in saved status
F72D but continue
F72E ---
F730 exit

F731 *****
***** MLI GET EOF CALL *****
*****
***** MLI GET EOF CALL *****
*****

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

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

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

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

F755 get the file entry <E593>
F758 ok? >>F79C
F75A no, bad path?
F75D no, real error >>F7B9
F75F else, make it VOL DIR type
F761 with name length = 0 (FE2A)
F766 no free blocks needed (FE5F)
F76C go through the motions to update the (FE5C)
F76F VCB block count. <E964>
F775 copy blocks free from VCB (D915)
F781 copy total blocks on volume to AUX_ID (D913)
F78F total - free = blocks used (FE5F)
F79C shift type down from high nibble (FE2A)
F7A8 copy the data to caller's parmlist (FDD7)
F7B9 and exit

```

```

ProDOS MLI -- V1.2 -- 6 SEP 86          NEXT OBJECT ADDR: F7B9          NEXT OBJECT ADDR: F832
-----
ADDR  DESCRIPTION/CONTENTS
-----

```

```

F7BA *****
***** MLI SET FILE INFO CALL *****
*****
F7BA get the file entry <E593>
F7BD error? >>F7E4
F7BF indicate backup needed now (BF95)
F7CE copy 13 parms from caller's list to (FDD7)
F7D1 file entry staging area >>F7D8
F7D8 ---
F7DD if any spurious access bits are on...
F7E1 access error!
F7E4 RETURN
F7E5 else, anything in his modification date?
F7E9 no >>F7EE
F7EB yes, go update directory >>E4C2
F7EE no, use system date then update directory >>E4B2
F7F1 *****
***** MLI RENAME CALL *****
*****
F7F1 follow path to file <E5A6>
F7F4 OK? >>F833
F7F6 no, bad name?
F7F8 no, real error >>F812
***** RENAME VOLUME ***
F7FA yes, copy new name <F917>
F7FD error? >>F812
F7FE get first length (D700)
F803 get next (D700)
F806 bad path if more than one name for vol >>F887
F80B files open on volume? (D911)
F80E no, continue >>F814
F810 yes, file open error
F812 ---
F813 RETURN
F814 make type/len for a VOL DIR HDR
F81B write new name to VOL HDR <F908>
F81E error? >>F889
F825 copy new name to device's VCB (D700)
F831 exit, no errors
F832 RETURN

```

```

***** RENAME FILE ***
F833 get path index <F925>
F836 copy old name with prefix to my buffer (D700)
F842 copy new name to buffer <F917>
F845 error? >>F889
F847 get path index <F925>
F84D compare all levels of names up to and (DC00)
F850 including the last. Find first which
F851 differ.
F855 save indicies into names which point to (FE84)
F858 final name. (FE85)
F85B ---
F865 exit if they match completely
F866 RETURN
F867 index to differing new name (FE84)
F86A point past it (D700)
F872 must be the last! (D700)
F875 it isn't >>F887
F877 it is, (FE85)
F87A do the same with the old name (DC00)
F885 difference is only in last index? >>F88B
F887 no, bad path error
F889 ---
F88A RETURN
F88B names good, follow path to new file <E5A6>
F88E better get an error >>F894
F890 if found, duplicate name in directory
F893 RETURN
F894 if error, better be file not found
F896 or else its really an error... >>F889
F898 copy old pathname again <E081>
F89B get its file entry <E593>
F89E error? >>F889
F8A0 search FCB's <EF9B>
F8A5 exit if the file is open for write >>F889
F8AA does ACCESS permit rename?
F8AC yes >>F8B2
F8AE no, access error
F8B0 ---
F8B1 RETURN
F8B2 get type/len from entry (FE2A)
F8B7 DIR file?
F8B9 yes, ok >>F8C3
F8BB seedling, sapling or tree?
F8BD yes, ok >>F8C3

```

ProDOS MLI -- V1.2 -- 6 SEP 86 NEXT OBJECT ADDR: F8BF

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

F8BF else, compatibility error  
 F8C3 copy new path again <F917>  
 F8C6 error? >>F889  
 F8C8 get length of last name (FE84)  
 F8D3 copy it and name to file entry buffer (D700)  
 F8E3 combine new len with type (D700)  
 F8E9 DIR file?  
 F8EB no, go update entry and exit >>F905  
 F8ED yes, (FE3B)  
 F8F3 read key block of this subdirectory <EBC9>  
 F8F6 error? >>F889  
 F8FB copy new name to DIR HDR (D700)  
 F900 and update directory's key block <F908>  
 F903 error? >>F889  
 F905 go update directory entry and exit >>E4C2

F908 \*\*\*\*\* COPY PATH TO BUFF & WRITE \*\*\*\*\*  
 F908 copy type/len and path to my buffer  
 F914 go write the block >>EBD5

F917 \*\*\*\*\* POINT TO NEW NAME \*\*\*\*\*  
 COPY TO BUFFER

F917 \$48/\$49 --> second pathname  
 F922 go copy it >>E08C

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 FCB if any <EF9B>  
 F93A FCB open? (FE62)  
 F93D yes, file open error >>F97C  
 F93F no free blocks needed  
 F947 go compute VCB free block count <E959>  
 F94A ok? >>F950  
 F94C error, disk full?  
 F94E no, real error >>F97E  
 F950 DESTROY enabled in ACCESS? (FE48)  
 F955 yes >>F95C  
 F957 no, access error

ProDOS MLI -- V1.2 -- 6 SEP 86 NEXT OBJECT ADDR: F95C

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

F95C check status of device (BF30)  
 F962 error? >>F97E  
 F964 point to key block (FE3B)  
 F973 DIR file?  
 F977 no >>F980  
 F979 yes, handle differently >>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 <EB76>  
 F9B5 error >>F97E  
 F9B7 update free block count in VCB <F9BD>  
 F9BA and go update the directory >>E4B2

\*\*\* SUBROUTINE TO UPDATE FREE BLOCK \*\*\*  
 \*\*\* COUNT IN VCB \*\*\*

F9BD 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)  
 F9EB read it <EBD9>  
 F9EE errors? >>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)  
 FAIL go back and pretend it's a seedling >>F997

```

ProDOS MLI -- V1.2 -- 6 SEP 86          NEXT OBJECT ADDR: FA13
-----
ADDR  DESCRIPTION/CONTENTS
-----

```

```

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

FA27  incompatible file format error

FA2C  ***** SET WRITE OCCURRED FLAG *****
FA2C  ***** TRUNCATE FILE AT EOF *****
FA2C  ***** TRUNCATE FILE AT EOF *****

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

FA3E  ***** TRUNCATE FILE AT EOF *****
FA3E  ***** TRUNCATE FILE AT EOF *****

FA3E  check storage type*16 (FE8C)
FA41  seedling?
FA43  yes >>FA52
FA45  no, sapling?
FA47  yes >>FA55
FA49  no, tree?
FA4B  yes >>FA58
FA4D  General error--wrong storage type
FA4F  jump to system death <EF0C>

FA52  go to seedling truncate >>FB24

FA55  go to sapling truncate >>FAEB

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

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

FA6A  copy up to 8 non-zero index bnock
FA6C  numbers to (DC00)
FA6F  a handy table (FE95)
FA80  ---
FA89  if there weren't 8 left to do, zero (FE95)
FA8C  remainder of the table (FE9D)

```

```

ProDOS MLI -- V1.2 -- 6 SEP 86          NEXT OBJECT ADDR: FA92
-----
ADDR  DESCRIPTION/CONTENTS
-----

```

```

FA92  ---
FA93  update master index counter (FE93)
FA98  for all 8 entries: (FE94)
FA9B  set BLKNUM, (FE95)
FAA3  (exit when a 0 entry is found), >>FA5D
FAAA  read the sub-index block, <EBD9>
FAAD  (quit if error), >>FABD
FAAF  zero all its blocks, <FB7D>
FAB2  (quit if error), >>FABD
FAB8  and loop until all 8 are done. >>FA98
FABA  then go back and reread master index >>FA5D
FABC  normal exit
FABD  RETURN

FABE  now go free all the sub-index blocks (FE8F)
FAC2  if error >>FABD
FAC5  write back master index <EBD5>
FAC7  if error >>FABD
FACA  EOF in first subindex? (FE8F)
FACC  if so, demote to sapling file >>FAE6
FAD1  else, BLKNUM = subindex block which (DC00)
FAD4  contains the EOF mark
FAD9  (exit if none there) >>FABC
FAE0  else, read the final subindex block <EBD9>
FAE3  and treat it as a sapling file >>FAF0
FAE5  unless there is an error.

FAE6  Demote tree to sapling <FB5B>
FAE9  if error >>FABD

*** TRUNCATE SAPLING FILE ***

FAEB  read index block <FB4F>
FAEE  if error >>FABD
FAF0  index of last block in the file (FE90)
FAF3  add one to point past end of file
FAF4  if zero, no blocks to free >>FB00
FAF6  zero blocks past EOF <FB7F>
FAF9  if error >>FABD
FAFB  write back modified index block <EBD5>
FAFE  if error >>FABD
FB00  index of last block in file (FE90)
FB03  this index block is empty! >>FB1A
FB05  get BLKNUM of last data block (DC00)
FB0D  (no block allocated?) >>FABC
FB14  read in last data block <EBD9>
FB17  and treat it as a seedling file >>FB29
FB19  unless error occurred.

```

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

NEXT OBJECT ADDR: FB19

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

NEXT OBJECT ADDR: FB9F

ADDR DESCRIPTION/CONTENTS

ADDR DESCRIPTION/CONTENTS

FB1A more index blocks (tree file)? (FE8F)  
 FB1D yes, must be tree file >>FB05  
 FB1F no, demote to seedling <FB5B>  
 FB22 if error >>FB4E

\*\*\* TRUNCATE SEEDLING FILE \*\*\*

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

FB4D exit normally  
 FB4E RETURN

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

FB4F Put index block number in A,X (FE8A)  
 FB55 Go read the block >>EBC9

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

FB5B get high byte of index block (FE8B)  
 FB5E and low byte (FE8A)  
 FB61 free the index block in the volume bitmap <EAL1A>  
 FB64 if error >>FB7C  
 FB66 Establish first block of old index block (DC00)  
 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 <EAL1A>  
 FB9A if error >>FBAB  
 FB9C Restore index to Y-reg (FE68)

FB9F zero this entry  
 FBA7 ---  
 FBAB loop through all entries >>FB85  
 FBAB save error message, if any  
 FBAD restore old BLKNUM  
 FBB3 and exit

FBBA \*\*\*\*\* ALLOCATE I/O BUFFER \*\*\*\*\*

FBBA ---  
 FBBA get I/O buffer page number  
 FBB9 can't be below \$800  
 FBBB else, error >>FBFF  
 FBBD can't be above \$BC00  
 FBBF else, error >>FBFF  
 FBC4 \$4A/\$4B --> I/O buffer  
 FBC8 must be page aligned! >>FBFF  
 FBCE ---  
 FBCE check each page of I/O buffer for <FC3A>  
 FBCF prior allocation in system bit map (BF58)  
 FBD2 ---  
 FBD2 if ok, mark each page as allocated <FC3A>  
 FBEE3 in system memory bit map (BF58)  
 FBEE4 assign buffer number (REFNUM\*2) in FCB (D800)  
 FBEE8 and save buffer location in buffer list  
 FBFD exit  
 FBFE RETURN  
 FBFF bad I/O buffer error  
 FC02 RETURN

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

FC03 ---  
 FC04 AREG contains buffer number \*2 (BF6E)  
 FC07 move buffer pointer to NEXTBUF variable (FEA8)  
 FC10 exit

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

FC11 is buffer already free? <FC03>  
 FC16 yes, exit >>FC38  
 FC1A zero its address in system global page (BF6F)  
 FC27 ---  
 FC28 free each page in buffer <FC3A>  
 FC2B by marking system bit map  
 FC38 exit  
 FC39 RETURN

ProDOS MLI -- V1.2 -- 6 SEP 86      NEXT OBJECT ADDR: FC39  
 ADDR    DESCRIPTION/CONTENTS

FC3A \*\*\*\*\* LOCATE BIT MAP POSITION \*\*\*\*\*  
 (GIVEN PAGE NUMBER)

FC3A XREG contains page number  
 FC3B compute page number times 8  
 FC3E use as offset for bitmask (FDCB)  
 FC45 page number / 8 = byte offset  
 FC46 into bitmap  
 FC48 exit

FC49 \*\*\*\*\* CHECK BUFFER VALIDITY \*\*\*\*\*  
 START > \$200    END < \$BF00

FC49 get buffer address (MSB)  
 FC4D must be >\$200 else error >>FBFF  
 FC4F get length (FEAG)  
 FC55 compute last page no. of buffer  
 FC5A ---  
 FC61 may not extend into \$BF00  
 FC63 else, error >>FBFF

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

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

FC75 \*\*\*\*\* MLI GET BUFF CALL \*\*\*\*\*  
 \*\*\*\*\* MLI GET BUFF CALL \*\*\*\*\*

FC75 get next available buffer  
 FC7A put its address in caller's parmlist  
 FC82 and exit  
 FC83 RETURN

FC84 \*\*\*\*\* MLI SET BUFF CALL \*\*\*\*\*  
 \*\*\*\*\* MLI SET BUFF CALL \*\*\*\*\*

FC84 mark his buffer allocated  
 FC89 error? >>FCAB  
 FC8B get old buffer address (FEA9)  
 FC95 free old buffer's pages in map <FC20>  
 FC9C copy old buffer contents  
 FC9E to new buffer  
 FCAA then exit

ProDOS MLI -- V1.2 -- 6 SEP 86      NEXT OBJECT ADDR: FCAB  
 ADDR    DESCRIPTION/CONTENTS

FCAB RETURN  
 FCAC \*\*\*\*\* GO TO QUIT CODE HANDLER \*\*\*\*\*

FCAC enable 2nd 4K bank of language card (C083)  
 (Quit code lives at \$D100-\$D3FF) (C083)  
 FCB4 Get first four bytes of page 0, (0000)  
 FCB7 save them on the stack  
 FCBB Set (\$00) -> \$D100  
 FCBD Set (\$02) -> \$1000  
 FCC9 Set Y = 0  
 FCCA 3 pages of code to copy  
 FCCC ---  
 FCCD copy quit code handler to \$1000

The next five lines of code were added for this version (1.2).  
 Fortunately they did not survive the next version (1.3).  
 Let's hope that whoever wrote this "fancy" code  
 is now working for Commodore.  
 pull 4 saved bytes off stack  
 and restore them to page 0 (FF04)  
 enable HIGH RAM BANK1 (C08B)  
 (MLI) (C08B)

FCDD point RESET vector at \$1000 (03F2)  
 FCE4 set power-up byte properly  
 FCF4 go to quit code handler at \$1000 >>1000  
 FCF9 ---

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

FCFC Get current P-reg in accumulator,  
 FCFE then save it on the stack  
 FCFE clear overflow flag  
 FD00 interrupts disabled?  
 FD02 no >>FD07  
 FD04 Yes, set overflow flag (FD25)  
 FD07 disable interrupts  
 FD08 enable RAM, BANK2 (C083)  
 FD0E set carry, indicating error  
 FD0F indicate 6 bytes to move to aux z-page  
 FD11 Call real driver thru page 3 routine <03D6>  
 FD14 store error number (BF0F)  
 FD17 enable RAM, BANK1 (C08B)

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

NEXT OBJECT ADDR: FD1D

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

NEXT OBJECT ADDR: FD46

ADDR DESCRIPTION/CONTENTS

ADDR DESCRIPTION/CONTENTS

FD1D restore original P-reg  
 FD1F 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, (C083)  
 FD29 execute the program there, <D400>  
 FD2C then back to BANK1 (C08B)  
 FD2F and return.

FD30 \*\*\*\*\* DATA AREA \*\*\*\*\*  
 \* DATA AREA \*  
 \*\*\*\*\*

FD30 \*\*\*\*\* MLI COMMAND TABLE \*\*\*\*\*  
 IN HASH CODE ORDER: IF COMMAND IS...  
 ABCD EFGH (IN BINARY BITS)  
 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

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 \*\*\*\*\* PARAMETER COUNT TABLE \*\*\*\*\*  
 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 \*\*\*\*\* MLI COMMAND ADDRESS TABLE \*\*\*\*\*

ProDOS MLI -- V1.2 -- 6 SEP 86      NEXT OBJECT ADDR: FD70  
 ADDR    DESCRIPTION/CONTENTS

```

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
| | DATETIME 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 0 1 0 - 0B
FDA4 0 0 1 - 0C
FDA5 0 0 1 - 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 -- V1.2 -- 6 SEP 86      NEXT OBJECT ADDR: FDAB  
 ADDR    DESCRIPTION/CONTENTS

```

FDAC ***** CONSTANTS - DATA AREA *****
FDAC    Blocks Used
FDAE    End of File
FDB1    Special ID (Must be 5 bits on)
FDB2    'HUSTONI'    Author's name
FDB9    Previous Block of Vol Dir Key Block
      ---
      THE FOLLOWING IS COPIED TO SUBDIR HDR+$20
FDBB    Version of ProDOS
FDBC    Minimum Version
FDBD    Access Byte    (D|Rn|B|000|W|R)
FDBE    Entry Length
FDBF    Entries per Block
FDC0    File Count
FDC2    Parent LSB (copied to SUBDIR HDR +$20)
      ---
FDC3    File Type (Directory)
FDC4    Block Number
FDC6    Number of Blocks
FDC8    End of File
  
```

FDCB \*\*\*\*\* BITMASK TABLE \*\*\*\*\*

```

FDCB 10000000
FDCC 01000000
FDCD 00100000
FDCE 00010000
FDCF 00001000
FDD0 00000100
FDD1 00000010
FDD2 00000001
  
```

FDD3 \*\*\*\*\* OFFSETS INTO FILE CONTROL BLOCKS \*\*\*\*\*  
 FDD3 \*\*\* (FCB's are at \$D803-\$D8FF) \*\*\*\*\*

```

FDD3    Key Block
FDD5    # Blocks Used
FDD7    End of File
  
```

FDDA \*\*\*\*\* SET/GET FILE\_INFO OFFSETS \*\*\*\*\*

```

FDDA    Access
FDDB    File Type
FDDC    Aux Type
FDDE    Storage Type
FDDF    Blocks Used (MSB on means GET only no SET)
  
```



ProDOS MLI -- V1.2 -- 6 SEP 86      NEXT OBJECT ADDR: FDE0  
 ADDR    DESCRIPTION/CONTENTS

ProDOS MLI -- V1.2 -- 6 SEP 86      NEXT OBJECT ADDR: FE4D  
 ADDR    DESCRIPTION/CONTENTS

FDE1    Datetime (Last Mod)  
 FDE5    Datetime (Creation)  
 FDE9    \*\*\*\*\* FATAL ERROR MESSAGE \*\*\*\*\*  
 FDE9    ' INSERT SYSTEM DISK AND RESTART -ERR 0 '  
 FE11    ---

FE4F    Header Pointer  
 FE51    \*\*\*\*\* Variable Work Area \*\*\*\*\*  
 FE51    3 Byte Scratch  
 FE54    ---  
 FE55    End of File  
 FE58    Previous Mark

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  
 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)

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  
 FE62    Number of Entries Examined or..  
 FE63    FCB already open flag  
 FE63    File Count  
 FE65    Entries/Block Loop Count/Free FCB's refnum  
 FE66    Free Entry Found Flag (if > 0) or..  
 FE66    # of 1st bitmap block with free bit on or..  
 FE67    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 \$80, needs writing)  
 FE71    Bitmap DEVNUM  
 FE72    Bitmap Block Number  
 FE74    Bitmap Block offset for Multiblock Bitmaps  
 FE75    New Mark to be Positioned to for Set Mark  
 FE75    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

```

ProDOS MLI -- V1.2 -- 6 SEP 86                NEXT OBJECT ADDR: FE7E
-----
ADDR  DESCRIPTION/CONTENTS
-----

```

```

FE7E MLI Command * 2
FE7F OKed into Access Flags ($20 - Backup)
FE80 Duplicate Volume Flag (if $FF)
FE81 Duplicate Volume's VCB index
FE82 MLI function code (low 5 bits)
      Characters in current Pathname indx lvl or
FE83 ONLINE: volname len - loop index
FE84 new pathname: index to last name
      Old pathname: index to last name or..
FE85 ONLINE: index to data buffer
FE86 Old PFXPTR value
FE87 Pathname fully qualified flag (if $FF)
      Pathname: temp save area for index or..
FE88 ONLINE: DEVCNT
FE89 close-all error code
FE8A Set EOF: new Key Block pointer
FE8C New storage type (SET_EOF)
FE8D Freed Blocks count
FE8F EOF Block number (MSB then LSB)
FE91 EOF byte offset into Block
FE93 EOF - Master index counter
FE94 Save area for index into table below

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
FE9D device table part two

FEA5 length of path, etc.
FEA8 next buffer address
FEAA 16 byte stack save area
FEBA 6 byte zero page save area
FEC0 Jump Vector, used for indirect jumps

FEC2 ***** $FEBF-$FEFF NOT USED *****
FEC2 not used

```

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

ProDOS MLI -- V1.3 -- 2 DEC 86      NEXT OBJECT ADDR: F9DD  
 ADDR    DESCRIPTION/CONTENTS

D700    MODULE STARTING ADDRESS

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

THE 1.3 VERSION OF THE PRODOS 8 MLI IS THE SAME AS VERSION 1.2, INSTRUCTION FOR INSTRUCTION, FROM THE START (\$DE00) TO ADDRESS \$F992. SOME BYTES BEFORE \$F992 CHANGE BECAUSE THEY REFER TO ADDRESSES GREATER THAN \$F992.

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

F980

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

```
F980 save the storage type (FEB5)
F987 set EOF to zero (FEB5)
F98D byte offset = $200
F992 "truncate" the file at EOF=0 <FA44>
F998 turn off the destroy flag (FEEB)
F99B error during truncation >>F97E
F9A3 free the key block in volume bitmap <EALA>
F9A6 error >>F97E
F9A8 mark the file as deleted in DIR
F9AD decrement file count in DIR (FE47)
F9B8 checkpoint volume bit map <EB76>
F9BB error >>F97E
F9BD update free block count in VCB <F9C3>
F9C0 and go update the directory >>E4B2
```

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

```
F9C3 add blocks freed to total free blocks (FE85)
F9C6 in VCB. (FEB6)
F9D8 start next search for free blocks at
F9DA start of bitmap. (D91C)
F9DD exit
```

\*\*\* DESTROY DIRECTORY FILE \*\*\*

```
F9DE DIR file?
F9E0 no, error >>FA2D
F9E2 read volume bitmap block <EB43>
F9E5 error? >>FA2C
F9E7 BLKNUM = key block pointer (FE64)
F9F1 read it <EBD9>
F9F4 errors? >>FA2C
F9F6 if DIR has any files... (DC25)
FA00 access error
FA05 write back block marking entry free (DC04)
FA0B error? >>FA2C
FA0D if "next_pointer" is zero.... (DC02)
FA17 go back and pretend it's a seedling >>F99D
FA19 else, (DC03)
FA1C free next block <EALA>
FA1F error? >>FA2C
FA21 BLKNUM = next block (DC02)
FA27 read it <EBC9>
FA2A if ok, continue in loop >>FA0D
FA2C else, error exit
```

FA2D 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 seedling?
FA49 yes >>FA58
FA4B no, sapling?
FA4D yes >>FA5B
FA4F no, tree?
FA51 yes >>FA5E
FA53 General error--wrong storage type
FA55 Jump to system death <BF0C>
FA58 go to seedling truncate >>FB2F
```

```

PRODOS MLI -- V1.3 -- 2 DEC 86      NEXT OBJECT ADDR: FA58      NEXT OBJECT ADDR: FAF0
ADDR  DESCRIPTION/CONTENTS
-----

```

```

FA5B go to sapling truncate >>FAF6
FA5E truncate tree,
FA60 at most 128 blocks in master index (FEBC)
FA63 read the master index <FB5A>
FA66 error? >>FAC8
FA68 at EOF yet? (FEBC)
FA6E yes >>FAC9

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

FA70 copy up to 8 non-zero index block
FA72 numbers to (DC00)
FA75 a handy table (FEBE)
FA86 ---
FA8F if there weren't 8 left to do, zero (FEBE)
FA92 remainder of the table (FEC6)
FA98 ---
FA99 update master index counter (FEBC)
FA9E for all 8 entries: (FEBD)
FAA1 set BLKNUM, (FEBE)
FAA9 (exit when a 0 entry is found), >>FA63
FAB0 read the sub-index block, <EBD9>
FAB3 (quit if error), >>FAC8
FAB5 zero all its blocks (if truncating) <FB95>
or swap pages (if destroying),
FAB8 (quit if error), >>FAC8
FABA write the former index block back out, <EBD5>
FAC3 and loop until all 8 are done. >>FA9E
FAC5 then go back and reread master index >>FA63
FAC7 normal exit
FAC8 RETURN

FAC9 now go free all the sub-index blocks (FEBC)
FACD which follow EOF <FB97>
FAD0 if error >>FAC8
FAD2 write back master index <EBD5>
FAD5 if error >>FAC8
FAD7 EOF in first subindex? (FEBC)
FADA if so, demote to sapling file >>FAF1
FADC else, BLKNUM = subindex block which (DC00)
FADF contains the EOF mark
FAF4 (exit if none there) >>FAC7
FAEB else, read the final subindex block <EBD9>
FAEE and treat it as a sapling file >>FAFB
FAF0 unless there is an error.

```

```

PRODOS MLI -- V1.3 -- 2 DEC 86      NEXT OBJECT ADDR: FAF0
ADDR  DESCRIPTION/CONTENTS
-----

```

```

FAF1 Demote tree to sapling <FB63>
FAF4 if error >>FAC8

*** TRUNCATE SAPLING FILE ***

FAF6 read index block <FB5A>
FAF9 if error >>FAC8
FAFB index of last block in the file (FEBC)
FAFE add one to point past end of file
FAFF if zero, no blocks to free >>FE0B
FB01 zero blocks past EOF (when truncating) <FB97>
or swap bytes for all but first block (when destroying).
FB04 if error >>FAC8
FB06 write back modified index block <EBD5>
FB09 if error >>FAC8
FB0B index of last block in file (FEBC)
FB0E this index block is empty! >>FB25
FB10 Get BLKNUM of last data block (DC00)
FB18 (no block allocated?) >>FAC7
FB1F read in last data block <EBD9>
FB22 and treat it as a seedling file >>FB34
FB24 unless error occurred.

FB25 more index blocks (tree file)? (FEBC)
FB28 yes, must be tree file >>FB10
FB2A no, demote to seedling <FB63>
FB2D if error >>FB59

*** TRUNCATE SEEDLING FILE ***

FB2F read key block <FB5A>
FB32 error? >>FB59
FB34 EOF in first page? (FEBC)
FB37 yes >>FB3F
FB39 EOF in second page?
FB3A no, exactly 256 bytes >>FB58
FB3C get byte offset (FEBA)
FB3F ---
FB41 zero bytes in second page (DD00)
FB47 EOF in first page? (FEBC)
FB4A no, we're done. >>FB55
FB4C yes, zero bytes in first page, too (FEBA)
FB55 then write block back and exit >>EBD5

FB58 exit normally
FB59 RETURN

```

ProDOS MLI -- V1.3 -- 2 DEC 86                   NEXT OBJECT ADDR: FB59  
 ADDR   DESCRIPTION/CONTENTS

ProDOS MLI -- V1.3 -- 2 DEC 86                   NEXT OBJECT ADDR: FBC7  
 ADDR   DESCRIPTION/CONTENTS

FB5A \*\*\*\*\* READ INDEX BLOCK \*\*\*\*\*  
 FB5A Put index block number in A,X (FB53)  
 FB60 Go read the block >>EBC9

FB63 \*\*\*\*\* DEMOTE FILE TO SMALLER FILE TYPE \*\*\*\*\*  
 FB63 get high byte of index block (FB64)  
 FB66 save it on stack  
 FB68 get low byte (FB63)  
 FB6B save it, too  
 FB6C free the index block in the volume bitmap <EA1A>  
 FB6F restore the block number of the index block  
 FB70 to zero page.  
 FB75 if error writing bitmap >>FB94  
 FB77 New index block is first block (DC00)  
 FB7A from old index block. (FB63)  
 FB83 For first entry in old index block,  
 FB85 zero the block number (if truncating) <FBC7>  
 or swap the bytes (if destroying).  
 FB89 reduce storage type by one (FB65)  
 FB91 Write the deleted index block back out. <EBD5>  
 FB94 RETURN

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

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

FB97 save BLKNUM  
 FB9D Save Y-register (index within block) (FE91)  
 FBAB if it is non-zero....  
 FBAB free the block in the volume bitmap <EA1A>  
 FB97 if error >>FBBE  
 FB97 Restore index to Y-reg (FE91)  
 FB97 zero this entry (when truncating) or <FBC7>  
 swap the two bytes (when destroying).  
 ---  
 FB97 loop through all entries >>FB9D  
 FB97 save error message, if any  
 FB97 restore old BLKNUM  
 FB97 and exit

FBC7 \*\*\*\*\* ZERO OR SWAP INDEX BLOCK BYTES \*\*\*\*\*  
 --- >>0001  
 FBC7 This a destroy operation? (FE6B)  
 FBCA Yes, swap the two bytes. >>FBCF  
 FBCB No, zero both bytes.  
 FBCD A 65C02 instruction!! >>FBD5  
 Note: we think this 65C02 instruction (BRA) snuck in by accident. It is the only 65C02 instruction in all of ProDOS and means the 1.3 version won't run on Apple IIs that don't have a 65C02. But it is easy to patch this--just change the byte at \$FBCD (\$4CCD in the load image) from \$80 to \$F0, 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)  
 FBDC RETURN

FBDD \*\*\*\*\* ALLOCATE I/O BUFFER \*\*\*\*\*  
 ---  
 FBDD get I/O buffer page number  
 FBE2 can't be below \$800  
 FBE4 else, error >>FC28  
 FBE6 can't be above \$BC00  
 FBE8 else, error >>FC28  
 FBED \$4A/\$4B --> I/O buffer  
 FBF1 must be page aligned! >>FC28  
 ---  
 FBE8 check each page of I/O buffer for <FC63>  
 FBEB prior allocation in system bit map (BF58)  
 FC08 ---  
 FC09 if ok, mark each page as allocated <FC63>  
 FC0C in system memory bit map (BF58)  
 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 *****
FC2D ---
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 >>FC61
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 (FDF4)
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 >>FC28
FC78 get length (FECF)
FC7E compute last page no. of buffer
FC83 ---
FC8A may not extend into $BF00
FC8C else, error >>FC28
      *** CHECK IF BLOCK OF MEMORY IS FREE ***
FC8F ---
FC90 see if this page is allocated <FC63>
FC96 if so, error >>FC28
FC98 else, check other paae also
FC9C then exit if both have been checked
FC9D RETURN

ProDOS MLI -- V1.3 -- 2 DEC 86          NEXT OBJECT ADDR: FC9D
-----
ADDR  DESCRIPTION/CONTENTS
-----
FC9E ***** MLI GET_BUFF CALL *****
      ***** MLI GET_BUFF CALL *****
FC9E get next available buffer
FC9E put its address in caller's parmlist
FC9E and exit
FC9E RETURN
FCAD ***** MLI SET_BUFF CALL *****
      ***** 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>
FCC5 copy old buffer contents
FCC7 to new buffer
FCD3 then exit
FCD4 RETURN
FCD5 ***** GO TO QUIT CODE HANDLER *****
FCD5 enable 2nd 4K bank of language card (C083)
FCDB (Quit code lives at $D100-$D3FF) (C083)
FCDD Get first four bytes of page 0, (0000)
FCE0 save them on the stack
FCE4 Set ($00) -> $D100
FCE6 Set ($02) -> $1000
FCF2 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
FD0D enable HIGH RAM BANK1 (C08B)
FD10 (MLI) (C08B)
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 $3EA,
      becomes the address of the device driver. Bytes $3E4 and $3E5
      are changed to the address of the real driver in aux high RAM.
      This routine must call the page 3 routine at $3D6
      because the MLI is in main high RAM and will be

```

ProDOS MLI -- V1.3 -- 2 DEC 86      NEXT OBJECT ADDR: FD25  
 ADDR    DESCRIPTION/CONTENTS

swapped out. The page 3 routine calls the real driver and returns here with the error code, if any.

FD25 Get current P-reg in accumulator,  
 FD27 then save it on the stack  
 FD28 clear overflow flag  
 FD29 interrupts disabled?  
 FD2B no >>FD30  
 FD2D yes, set overflow flag (FD4E)  
 FD30 disable interrupts  
 FD31 enable RAM, BANK2 (C083)  
 FD37 set carry, indicating error  
 FD38 indicate 6 bytes to move to aux z-page  
 FD3A Call real driver thru page 3 routine <03D6>  
 FD3D store error number (BF0F)  
 FD40 enable RAM, BANK1 (C08B)  
 FD46 restore original P-reg  
 FD48 if error number is zero, (BF0F)  
 FD4B then indicate no error; >>FD4E  
 FD4D otherwise indicate error  
 FD4E RETURN

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

FD4F Switch to BANK2 of high RAM, (C083)  
 FD52 execute the program there, <D400>  
 FD55 then back to BANK1 (C08B)  
 FD58 and return.

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

FD59 \*\*\*\*\* MLI COMMAND TABLE \*\*\*\*\*  
 IN HASH CODE ORDER: IF COMMAND IS...  
 ABCD EFGH (IN BINARY BITS)  
 INDEX IS COMPUTED AS:  
 000D EFGH  
 +0000 ABCD

FD59 GET BUF  
 FD5A UNUSED  
 FD5B UNUSED  
 FD5C UNUSED  
 FD5D ALLOC INTERRUPT

ProDOS MLI -- V1.3 -- 2 DEC 86      NEXT OBJECT ADDR: FD5E  
 ADDR    DESCRIPTION/CONTENTS

FD5E DEALLOC INTERRUPT  
 FD60 UNUSED  
 FD61 READ BLOCK  
 FD62 WRITE BLOCK  
 FD63 GET TIME  
 FD64 EXIT  
 FD65 CREATE  
 FD66 DESTROY  
 FD67 RENAME  
 FD68 SET FILE INFO  
 FD69 GET FILE INFO  
 FD6A ON LINE  
 FD6B SET PREFIX  
 FD6C GET PREFIX  
 FD6D OPEN  
 FD6E NEWLINE  
 FD6F READ  
 FD70 WRITE  
 FD71 CLOSE  
 FD72 FLUSH  
 FD73 SET MARK  
 FD74 GET MARK  
 FD75 UNUSED  
 FD76 SET EOF  
 FD77 GET EOF  
 FD78 SET BUF

FD79 \*\*\*\*\* PARAMETER COUNT TABLE \*\*\*\*\*

FD79 GET BUF  
 FD7A UNUSED  
 FD7B UNUSED  
 FD7C UNUSED  
 FD7D ALLOC INTERRUPT  
 FD7E DEALLOC INTERRUPT  
 FD80 UNUSED  
 FD81 READ BLOCK  
 FD82 WRITE BLOCK  
 FD83 GET TIME  
 FD84 EXIT  
 FD85 CREATE  
 FD86 DESTROY  
 FD87 RENAME  
 FD88 SET FILE INFO  
 FD89 GET FILE INFO  
 FD8A ON LINE  
 FD8B SET PREFIX  
 FD8C GET PREFIX  
 FD8D OPEN  
 FD8E NEWLINE  
 FD8F READ

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

NEXT OBJECT ADDR: FD90

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

NEXT OBJECT ADDR: FDCC

ADDR DESCRIPTION/CONTENTS

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  
 FDA1 GET FILE INFO  
 FDA3 ON LINE  
 FDA5 SET PREFIX  
 FDA7 GET PREFIX  
 FDA9 OPEN  
 FDAB NEWLINE  
 FDAD READ  
 FDAF WRITE  
 FDB1 CLOSE  
 FDB3 FLUSH  
 FDB5 SET MARK  
 FDB7 GET MARK  
 FDB9 SET EOF  
 FDBB GET EOF  
 FDBD SET BUF  
 FDBF GET BUF

FDCC \*\*\*\*\* BITMASK TABLE \*\*\*\*\*

FDCC 10000000  
 FDC2 01000000  
 FDC3 00100000  
 FDC4 00010000  
 FDC5 00001000  
 FDC6 00000100  
 FDC7 00000010  
 FDC8 00000001

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

FD90 1 0 1 - 00  
 FDC1 1 0 1 - 01  
 FDC2 1 0 1 - 02  
 FDC3 1 0 1 - 03  
 FDC4 1 0 0 - 04  
 FDC5 1 0 0 - 05  
 FDC6 0 0 0 - 06  
 FDC7 0 0 0 - 07  
 FDC8 0 0 0 - 08  
 FDC9 1 0 0 - 09  
 FDCA 0 1 0 - 0A  
 FDCE 0 1 0 - 0A

FDD5 \*\*\*\*\* CONSTANTS - DATA AREA \*\*\*\*\*

FDD5 Blocks Used  
 FDD7 End of File  
 FDDA Special ID (Must be 5 bits on)  
 Fddb 'HUSTON!' 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|000|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 10000000  
 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



PRODOS MLI -- V1.3 -- 2 DEC 86 NEXT OBJECT ADDR: FDFD

ADDR DESCRIPTION/CONTENTS

```

FDFF # Blocks Used
FE00 End of File
FE03 ***** SET/GET FILE_INFO OFFSETS *****
FE04 Access
FE04 File Type
FE05 Aux Type
FE07 Storage Type
FE08 Blocks Used (MSB on means GET only no SET)
FE0A Datetime (Last Mod)
FE0E Datetime (Creation)
FE12 ***** FATAL ERROR MESSAGE *****
FE12 INSERT SYSTEM DISK AND RESTART -ERR 0
FE3A ---
FE3A ***** VARIABLES - DATA AREA *****
FE3A Parent Pointer Block
FE3C Parent Entry Number
FE3D Parent Entry Length
FE3E Datetime (Creation)
FE42 Version
FE43 Min Version
FE44 Access Byte
FE45 Entry Length
FE46 Entries per Block
FE47 File Count
FE49 Bit Map Pointer
FE4B Total Blocks
    THE FOLLOWING 6 BYTES UNIQUELY IDENTIFY
    A FILE:
FE4D Device Number
FE4E Current Directory Block Number (HDR)
FE50 Block Number of File Entry in Directory
FE52 File Entry Number in Directory
FE53 ***** FILE ENTRY BUFFER *****
FE53 Type/Length (TTTTLLLL)
FE54 File Name (Max 15) >>000F
FE63 File Type
FE64 Key Pointer
FE66 Blocks Used
    
```

PRODOS MLI -- V1.3 -- 2 DEC 86 NEXT OBJECT ADDR: FE68

ADDR DESCRIPTION/CONTENTS

```

FE68 End of File
FE6B Datetime (Creation)
FE6F Version
FE70 Min Version
FE71 Access Attribute
FE72 Aux Type (Load Address/Record Length)
FE74 Datetime (Last Mod)
FE78 Header Pointer
FE7A ***** Variable Work Area *****
FE7A 3 Byte Scratch
FE7D ---
FE7E End of File
FE81 Previous Mark
FE84 Compare Vol Name Scratch
FE85 Offset into VCB Table ($D900)
FE86 Offset into FCB Table ($D800)
FE87 Free FCB found Flag
FE88 Number of Free Blocks needed
FE8A Storage Type
    Number of Entries Examined or..
FE8B FCB already open flag
FE8C File Count
FE8E Entries/Block Loop Count/Free FCB's refnum
    Free Entry Found Flag (if > 0) or..
    # of 1st bitmap block with free bit on or..
FE8F bit for free
FE90 # Blocks in Bitmap left to search
FE91 Y Register temp
FE92 Pathname Length
FE93 Devnum for Prefix Directory Header
FE94 Block of Prefix Directory Header
FE96 Bitmap Byte Offset in Page
FE97 Bitmap Page Offset
FE98 Bitmap Buffer Page (0 or 1)
FE99 Bitmap Flag (if $80, needs writing)
FE9A Bitmap DEVNUM
FE9B Bitmap Block Number
FE9D Bitmap Block Offset for Multiblock Bitmaps
    
```

ProDOS MLI -- V1.3 -- 2 DEC 86	Next Object Addr: FE9D	ProDOS MLI -- V1.3 -- 2 DEC 86	Next Object Addr: FEEB
ADDR DESCRIPTION/CONTENTS		ADDR DESCRIPTION/CONTENTS	
FE9E	New Mark to be Positioned to for Set Mark or New Moving Mark (for READ) or New EOF for SET_EOF	FEEC	***** \$FEEC-\$FEFF NOT USED ***** FEEC not used
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 (\$20 - Backup)		
FEA9	Duplicate Volume Flag (if \$FF)		
FEAA	Duplicate Volume's VCB index		
FEAB	MLI function code (low 5 bits)		
	Characters in current Pathname indx 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)		
FEFE	device table part one		
FEC6	device table part two		
FECE	length of path, etc.		
FED1	next buffer address		
FED3	16 byte stack save area		
FEF3	6 byte zero page save area		
FEF9	---		
FEF9	Jump Vector, used for indirect jumps		
FEFB	Destroy flag (1 = destroy operation)		

ProDOS System Global Page -- V1.2 NEXT OBJECT ADDR: 3F00

ADDR DESCRIPTION/CONTENTS

BF00 MODULE STARTING ADDRESS  
 \*\*\*\*\*  
 \* PRODOS SYSTEM GLOBAL PAGE \*  
 \* \*  
 \* \* VERSION 1.2 -- 6 SEP 86 \*  
 \* \* VERSION 1.3 -- 2 DEC 86 \*  
 \* \*  
 \*\*\*\*\*

BF00 \*\*\*\*\* MLI AND IRQ HANDLER EQUATES \*\*\*\*\*  
 DE00 Main MLI entry point.  
 DEAC 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 JMP to MLI. >>BF4B  
 BF03 JSPARE System death address. >>BF03  
 BF06 DATETIME JMP To Date/Time routine (RTS if no clock).  
 BF07 Normal clock code address.  
 BF09 SYSERR JMP to system error handler. >>DEFF  
 BF0C SYSDEATH JMP to system death handler. >>E009  
 BF0F SERR System error number.

BF10 \*\*\*\*\* DEVICE INFORMATION \*\*\*\*\*  
 BF10 DEVADR01 Slot 0 reserved.  
 BF12 DEVADR11 Slot 1, drive 1 device driver address.  
 BF14 DEVADR21 Slot 2, drive 1 device driver address.  
 BF16 DEVADR31 Slot 3, drive 1 device driver address.  
 BF18 DEVADR41 Slot 4, drive 1 device driver address.  
 BF1A DEVADR51 Slot 5, drive 1 device driver address.  
 BF1C DEVADR61 Slot 6, drive 1 device driver address.  
 BF1E DEVADR71 Slot 7, drive 1 device driver address.  
 BF20 DEVADR02 Slot 0 reserved.  
 BF22 DEVADR12 Slot 1, drive 2 device driver address.  
 BF24 DEVADR22 Slot 2, drive 2 device driver address.  
 BF26 DEVADR32 /RAM device address (128K required).  
 BF28 DEVADR42 Slot 4, drive 2 device driver address.  
 BF2A DEVADR52 Slot 5, drive 2 device driver address.  
 BF2C DEVADR62 Slot 6, drive 2 device driver address.  
 BF2E DEVADR72 Slot 7, drive 2 device driver address.

ProDOS System Global Page -- V1.2 NEXT OBJECT ADDR: BF2E

ADDR DESCRIPTION/CONTENTS

BF30 DEVNUM Slot and drive (DSSS0000) of last device.  
 BF31 DEVCNT Count (minus 1) of active devices  
 BF32 DEVLST List of active devices (slot, drive, and  
 identification--DSSSIIII).

BF40 "(C)APPLE'83" Copyright notice.  
 BF4B \*\*\*\*\* PATCHES TO ORIGINAL GLOBAL PAGE \*\*\*\*\*

BF4B MLIENTL1 Push status reg on stack.  
 BF4C Disable interrupts  
 BF4D GO to MLIENTL. >>BF07  
 BF50 AFTIRQ Read high RAM, BANK1 (C08B)  
 BF53 and continue IRQ exit. >>FFD8  
 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 \*\*\*\*\*  
 BF80 INTRUPT1 Interrupt handler address (highest priority).  
 BF82 INTRUPT2 Interrupt handler address.  
 BF84 INTRUPT3 Interrupt handler address.  
 BF86 INTRUPT4 Interrupt handler address (lowest priority).  
 BF88 INTAREG A-register savearea.  
 BF89 INTXREG X-register savearea.  
 BF8A INTYREG Y-register savearea.  
 BF8B INTSREG S-register savearea.  
 BF8C INTPREG P-register savearea.  
 BF8D INTBANKID Bank ID byte (ROM, RAM1, or RAM2).  
 BF8E INTADDR Interrupt return address.

BF90 \*\*\*\*\* GENERAL SYSTEM INFO \*\*\*\*\*  
 BF90 DATE YYYYYYMM MMMDDDD.  
 BF92 TIME ...HHHHH ..MMMMMM.  
 BF94 LEVEL Current file level.  
 BF95 BUBIT Backup bit.  
 BF96 SPARE1 Currently unused.  
 BF98 MACHID Machine ID byte.  
 00... 0... II

ProDOS System Global Page -- V1.2      NEXT OBJECT ADDR: BF99  
 -----  
 ADDR    DESCRIPTION/CONTENTS  
 -----

01.. 0... II+  
 10.. 0... Iie 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  
 .... ..00 No compatible clock  
 .... ..01 Compatible clock present

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

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

BFA0 EXIT    \$E000 same as save byte? (E000)  
 BFA3    Yes, check for BANK1/BANK2 >>BFAA  
 BFA5    No, enable ROM (C082)  
 BFA8    and exit now. >>BFB5  
 BFAB    Get RAM save byte for \$D000. (BFF5)  
 BFB0    Is it the same as BANK1? (D000)  
 BFB2    Yes, exit now. >>BFB5  
 BFB5    No, switch to BANK2. (C083)  
 BFB6    Restore A-Register and  
       return from the interrupt.

BF7 \*\*\*\*\* MLI ENTRY, CONTINUED \*\*\*\*\*  
 BF7 MLICONT    Carry set will  
 BFB8    roll flag bit into MLIACTV. (BF9B)  
 BFB9    Save high memory configuration (E000)  
 BFB0    by storing \$E000 in BNKBYT1 (BFF4)  
 BFC1    and \$D000 (D000)  
 BFC4    in BNKBYT2. (BFF5)  
 BFC7    Then enable high RAM, BANK1 (C08B)  
 BFCA    for read and write. (C08B)  
 BFCD    Jump to actual MLI. >>DE00

ProDOS System Global Page -- V1.2      NEXT OBJECT ADDR: BFCD  
 -----  
 ADDR    DESCRIPTION/CONTENTS  
 -----

BFD0 \*\*\*\*\* INTERRUPT ROUTINES \*\*\*\*\*  
 BFD0 IRQIT    Determine state of high memory. (BF8D)  
 BFD3    High RAM, BANK1 enabled. >>BFE2  
 BFD5    High RAM, BANK2 enabled. >>BFDF  
 BFD7    System have only 48K?  
 BFD8    Yes, only ROM in high memory. >>BFE7  
 BFDA    Enable ROM. (C081)  
 BFDD    Always branch. >>BFE7  
 BFDF    Switch to BANK2. (C083)  
 BFE2    Preset BANKID for ROM.  
 BFE4    (Later reset if high RAM interrupt). (BF8D)  
 BFE7    Restore accumulator and  
 BFEA    exit.

BFE8    Enable high RAM, BANK1 (C08B)  
 BFE9    for read and write. (C08B)  
 BFEA    Jump into MLI. >>DF4E

BFF4 \*\*\*\*\* DATA \*\*\*\*\*  
 BFF4 BNKBYT1    Storage for byte at \$E000.  
 BFF5 BNKBYT2    Storage for byte at \$D000.  
 BFF6    \$BFF6-\$BFFB currently not used.

BFFC \*\*\*\*\* VERSION INFORMATION \*\*\*\*\*  
 BFFC IBKVER    Minimum version of Kernel needed for this interpreter.  
 BFFD IVERSION    Version number of this in interpreter.  
 BFFE KBKVER    Currently undefined. Reserved for future use.  
 BFFF KVERSION    MLI Version number:  
               =2 in Version 1.2  
               =3 in Version 1.3

ProDOS System Global Page -- V1.2      NEXT OBJECT ADDR: BFCD  
 -----  
 ADDR    DESCRIPTION/CONTENTS  
 -----

ProDOS QUIT Code -- V1.2 -- 6 SEP 86      NEXT OBJECT ADDR: 1000  
 ADDR    DESCRIPTION/CONTENTS

ProDOS QUIT Code -- V1.2 -- 6 SEP 86      NEXT OBJECT ADDR: 1000  
 ADDR    DESCRIPTION/CONTENTS

1000    MODULE STARTING ADDRESS  
 \*\*\*\*\*  
 \*    \* QUIT Code  
 \*    \* Stored in BANK2 of High RAM  
 \*    \* at \$D100. A move routine in  
 \*    \* the MLI moves the code to  
 \*    \* \$1000 and Jumps to it when a  
 \*    \* QUIT command is issued.  
 \*    \*  
 \*    \* VERSION 1.2 -- 6 SEP 86  
 \*    \* Move routine at \$FCA9  
 \*    \*  
 \*    \* VERSION 1.3 -- 2 DEC 86  
 \*    \* Move routine at \$FCD5  
 \*    \*  
 \*\*\*\*\*

FF3A    Sound Bell  
 1000    \*\*\*\*\* INITIALIZATION \*\*\*\*\*  
 1000    Select ROM (C082)  
 1003    Disable 80 column card (C00C)  
 1006    Select standard character set            (C00E)  
 1009    Clear 80-column store                    (C000)  
 100C    Set Normal display (white on black) <FE84>  
 100F    Initialize 40-column display <FE84>  
 1012    Set Video <FE93>  
 1015    Set Keyboard <FE89>  
 1018    \*\*\*\*\* INITIALIZE MEMORY BITMAP \*\*\*\*\*  
 1018    Mark pages \$0, \$1, \$4 through \$7  
 101A    and \$BF as in use

1000    \*\*\*\*\* ZERO PAGE EQUATES \*\*\*\*\*  
 0024    Cursor Horizontal  
 0025    Cursor Vertical  
 1000    \*\*\*\*\* EXTERNAL EQUATES \*\*\*\*\*  
 0280    Prefix Buffer  
 1800    Buffer  
 2000    Buffer  
 BF00    MLI Entry  
 BF58    Bitmap  
 1000    \*\*\*\*\* SOFT SWITCHES \*\*\*\*\*  
 C000    Keyboard  
 C000    Disable 80 column store  
 C00C    Disable 80 column card  
 C00E    Select standard character set  
 C082    ROM select

102D    \*\*\*\*\* DISPLAY CURRENT PREFIX \*\*\*\*\*  
 102D    Clear Screen and Home cursor <FC58>  
 1030    Go down 1 line <FD8E>  
 1033    Point to prompt number 1  
 1035    and print it out <11D6>  
 1038    Position to line 3  
 103F    Call MLI (GET PREFIX) <BF00>  
 1042    Data: GET\_PREFIX command number  
 1043    Data: Pointer to Parameter list  
 1045    Get length of Prefix (0280)  
 1048    Put a 0  
 104A    at the end of the Prefix (0281)  
 104D    Check prefix length.. (0280)  
 1050    If length=0, there is no current Prefix >>105D  
 1052    If non-zero, display the current Prefix (0280)  
 1057    on the video screen (05FF)

1000    \*\*\*\*\* MONITOR EQUATES \*\*\*\*\*  
 FE84    Initialize 40-column display  
 FC58    Home  
 FC9C    Clear to end of line  
 FD0C    Read a key  
 FD8E    Output a Carriage Return  
 FDED    Output a Character  
 FE84    Set Normal display  
 FE89    Set Keyboard  
 FE93    Set Video

105D    \*\*\*\*\* GET PREFIX NAME \*\*\*\*\*  
 105D    Initialize counter  
 1064    Read a key <FD0C>  
 1067    Is it CARRIAGE RETURN?  
 1069    Yes, then accept Prefix >>10BD  
 106B    No, then save character  
 106C    Clear to end of line <FC9C>  
 106F    Retrieve character  
 1070    Is it ESCAPE?  
 1072    Yes, start all over again >>102D  
 1074    Is it CANCEL?  
 1076    Yes, start all over again >>102D  
 1078    Is it TAB?  
 107A    Yes, sound Bell, get another character >>1093

ProDOS QUIT Code -- V1.2 -- 6 SEP 86	NEXT OBJECT ADDR: 107C	ProDOS QUIT Code -- V1.2 -- 6 SEP 86	NEXT OBJECT ADDR: 10E5
ADDR	DESCRIPTION/CONTENTS	ADDR	DESCRIPTION/CONTENTS
107C	Is it DELETE?	10E5	Initialize counter
107E	Yes. >>1084	10E7	Wait for keypress <FD0C>
1080	Is it BACKSPACE?	10EA	Is it ESCAPE?
1082	No, keep checking >>1091	10EC	No, keep checking >>10F4
1084	Yes, is there room to move back?	10EE	Yes, get Cursor horizontal position
1086	No, don't try >>108B	10F0	If not 0 try again >>10D3
1088	Decrement cursor horizontal position	10F2	If 0 start all over again >>10D1
108A	Decrement counter	10F4	Is it CANCEL?
108B	Clear to end of line <FC9C>	10F6	Yes, try again >>10D3
108E	Try again >>1064	10F8	Is it TAB?
1091	Continue if greater or equal to BACKSPACE >>1099	10FA	Yes, sound Bell - try again >>1109
1093	Else, sound Bell <FF3A>	10FC	Is it DELETE?
1096	Try again >>1064	10FE	Yes >>1104
1099	Is it less than or equal to "Z"?	1100	Is it BACKSPACE?
109B	Yes, keep checking >>109F	1102	No, keep checking >>1107
109D	Turn off lowercase	1104	Yes, then handle it >>110C
109F	Is it less than ". "?	1107	Continue if greater than BACKSPACE >>110F
10A1	Yes, Invalid - try again >>1093	1109	Sound Bell <FF3A>
10A3	Is it greater than "Z"?	110C	Go back and try again >>10E7
10A5	Yes, Invalid - try again >>1093	110F	Is it CARRIAGE RETURN?
10A7	Is it less than or equal to "9"?	1111	Yes, then go load Application >>113C
10A9	Yes, keep checking >>10AF	1113	Is it less than or equal to "Z"?
10AB	Is it less than "A"?	1115	Yes, keep checking >>1119
10AD	Yes, Invalid - try again >>1093	1117	Turn off lower case
10AF	Else, valid character - increment counter	1119	Is it less than ". "?
10B0	Found 39 characters?	111B	Yes, Invalid - try again >>1109
10B2	Yes, then start all over >>1076	111D	Is it greater than "Z"?
10B4	Put valid character in buffer (0280)	111F	Yes, Invalid - try again >>1109
10B7	and Print it <FDED>	1121	Is it less than or equal to "9"?
10BA	Go back for more >>1064	1123	Yes, keep checking >>1129
10BD	Check counter	1125	Is it less than "A"?
10BF	If 0 then go on >>10D3	1127	Yes, Invalid - try again >>1109
10C1	Else, save length (0280)	1129	Else, valid character - save it
10C4	Call MLI (SET PREFIX) <BF00>	112A	Clear to end of line <FC9C>
10C7	Data: SET PREFIX command number	112D	Retrieve character
10C8	Data: Pointer to Parameter list	112E	and Print it <FDED>
10CA	Carry on if no error >>10D3	1131	Increment counter
10CC	Sound Bell <FF3A>	1132	Found 39 characters?
10CF	Force branch to	1134	Yes, start again >>10F6
10D1	always be taken >>1076	1136	No, save character in buffer (0280)
10D3	***** GET APPLICATION NAME *****	1139	and go get another >>10E7
10D3	***** EXECUTE APPLICATION *****	113C	***** LOAD AND EXECUTE APPLICATION *****
10D3	Clear Screen and Home cursor <FC58>	113C	Output a blank
10D6	Go down 1 line <FD8E>	1141	Store length of Application name (0280)
10D9	Point to prompt number 2	1144	Call MLI (GET FILE INFO) <BF00>
10DB	and print it out <11D6>	1147	Data: GET FILE INFO command number
10DE	Position to line 3	1148	Data: Pointer to Parameter list

ProDOS QUIT Code -- V1.2 -- 6 SEP 86      NEXT OBJECT ADDR: 114A      NEXT OBJECT ADDR: 11BA

ADDR    DESCRIPTION/CONTENTS

```

114A Continue if no error >>114F
114C Else, go to Error Handler >>11E2

114F Get File Type (12A5)
1152 Is it ProDOS System file?
1154 Yes, continue >>115B
1156 No, indicate Error $01
1158 Go to Error Handler >>11E2

115B Set Reference number to 0
1160 Call MLI (CLOSE) <BF00>
1163 Data: CLOSE command number
1164 Data: Pointer to Parameter list
1166 Continue if no error >>116B
1168 Else, go to Error Handler >>11E2
116B Get Access Byte (12A4)
1170 Yes, >>1177
1172 No, Indicate Error $27
1174 Go to Error Handler >>11E2

1177 Call MLI (OPEN) <BF00>
117A Data: OPEN command number
117B Data: Pointer to Parameter list
117D Continue if no error >>1182
117F Else, go to Error Handler >>11E2

1182 Get Reference Number (12B8)
1185 and update READ and (12BC)
1188 GET EOF parameter lists (12C4)
118B Call MLI (GET_EOF) <BF00>
118E Data: GET_EOF command number
118F Data: Pointer to Parameter list
1191 If error, handle it >>11E2

1193 Is EOF mark less than $10000 (12C7)
1196 Yes, continue on >>119C
1198 No, Indicate Error $27
119A Go to Error Handler >>11E2

119C Transfer EOF to Request count (12C5)
119F in READ parameter list (12BF)
11A8 Call MLI (READ) <BF00>
11AB Data: READ command number
11AC Data: Pointer to Parameter list
11AE Save status of READ
11AF Call MLI (CLOSE) <BF00>
11B2 Data: Get Prefix command number
11B3 Data: Pointer to Parameter list
11B5 Continue if no error >>11BB
11B7 Else, retrieve status
11B8 and go to Error Handler >>11E2

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

11BA Was READ good?
11BB No, go to Error Handler >>11B7
11BD Yes, execute application >>2000

11C0 ***** BACKSPACE ROUTINE *****
11C0 Get cursor position horizontal
11C2 If 0 exit routine >>11D3
11C4 Decrement counter
11C5 Output a space
11CA Move cursor back 2 spaces
11CE Output a space <FDED>
11D1 Move cursor back 1 space
11D3 Return to get another character >>10E7

11D6 ***** PRINT TEXT ROUTINE *****
11D6 Get a character (1211)
11D9 If it is 0 then exit >>11E1
11DB Output it <FDED>
11DE Increment offset
11DF Get another character >>11D6
11E1 Return to caller

11E2 ***** PRINT ERROR MESSAGE *****
11E2 Save Accumulator (Error Number)
11E4 Position to line 12
11EB Get Error number
11ED Is it 1?
11EF No, keep checking >>11F5
11F1 Yes, point to error message 1
11F3 and go print it >>120B
11F5 Is it $40?
11F7 Yes, print error message 3 >>1209
11F9 Is it $44?
11FB Yes, print error message 3 >>1209
11FD Is it $45?
11FF Yes, print error message 3 >>1209
1201 Is it $46?
1203 Yes, print error message 3 >>1209
1205 Point to error message 2
1207 and go print it >>120B

```

ProDOS QUIT Code -- V1.2 -- 6 SEP 86      NEXT OBJECT ADDR: 1209  
 -----  
 ADDR    DESCRIPTION/CONTENTS

1209 Point to error message 3  
 120B Print Error message <llD6>  
 120E Get application name again >>1ØDE

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

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

    Prompt2  
 1239 'ENTER PATHNAME OF NEXT APPLICATION'

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

    Error2  
 1273 Ring Bell  
 1274 'I/O ERROR'

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

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

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

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

    CLOSE Parmlist  
 12B9 Parmcount  
 12BA Reference Number

ProDOS QUIT Code -- V1.2 -- 6 SEP 86      NEXT OBJECT ADDR: 12BA  
 -----  
 ADDR    DESCRIPTION/CONTENTS

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

    GET\_EOF Parmlist  
 12C3 Parmcount  
 12C4 Reference Number  
 12C5 EOF Mark

    GET/SET\_PREFIX Parmlist  
 12C8 Parmcount  
 12C9 Pathname

12CB \*\*\*\*\* \$12CB-\$12FF UNUSED \*\*\*\*\*  
 12CB These unused bytes are \$D3CB-\$D3FF in high RAM  
 12FF and \$5BCB-\$5BFF when loaded as part of "PRODOS" file.



Disk II Device Driver -- V1.2 -- 6 SEP 86 NEXT OBJECT ADDR: D000

ADDR DESCRIPTION/CONTENTS

```

D000 MODULE STARTING ADDRESS
*****
*
* 5.25" DISK DEVICE DRIVER
*
* RESIDES AT $D000-$D6FF
*
* VERSION 1.2 -- 6 SEP 86
* (SAME AS IN VERSION 1.1.1)
*
*****
D000 ***** ZERO PAGE EQUATES *****
003A Checksum
003A Workbyte
003E Slot (Temporary)
0042 Command
0043 Unit Number
0044 I/O Buffer Pointer (low)
0045 I/O Buffer Pointer (high)
0046 Block Number (low)
0047 Block Number (high)

```

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

```

1000 Dummy Block Buffer (1st half)
1100 Dummy Block Buffer (2nd half)
D000 ***** EXTERNAL EQUATES *****
C080 Phase Zero Off
C088 Motor Off
C089 Motor On
C08A Drive Select
C08C Read Data Register
C08D Write Data Register
C08E Set Read Mode
C08F Set Write Mode
C0EC Read Data Register (slot 6)

```

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

```

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

```

Disk II Device Driver -- V1.2 -- 6 SEP 86 NEXT OBJECT ADDR: D010

ADDR DESCRIPTION/CONTENTS

```

D010 ---
D014 00000000T TTTTABC
D015 . . . >>D010
D017 . . . >>D01C
D018 . . . >>D01C
D01A 00TTTTT 0000BC0A
D01C ---
D020 Preserve Sector Number
D021 Execute command <D038>
D024 Restore Sector Number - Was prior action OK?
D025 NO, then exit >>D030
D027 Increment Buffer Pointer
D029 Increment Sector Number by 2 for rest of Block
D02B Execute command <D038>
D02E Decrement Buffer Pointer (to start of block)
D030 Get error number (if any - 0 indicates no error) (D358)
D033 Return to caller

```

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

```

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

```

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

```

D038 Set recalibration count to 1
D03D Preserve sector number (D357)
D040 Get "Unitnum" DSS0000
D042 Strip out Drive 0SS0000
D044 Preserve slot number
D046 Check for slot change, turn off motor if so <D69B>
D049 See if motor is on <D4DA>
D04C Save test results
D04F Initialize counter for delay routine (D370)
D054 See if slot or drive has changed (D359)
D057 Update "current" unit number (D359)
D05A Save test results
D05B Put drive number in Carry flag
D05C Turn motor on (C089)
D062 Select appropriate drive (C08A)
D065 Check test results - Same slot/drive?
D066 Yes, then skip delay >>D072
D069 Wait for new Drive
D06B to come up to speed <D385>
D072 Is command a status request?
D074 Yes, then do not move disk arm >>D07C
D076 Get track number for current request (D356)
D079 And go there <D10C>
D07C Check test results - Was motor on?
D07D Yes, then skip delay >>D08E

```

Disk II Device Driver -- V1.2 -- 6 SEP 86 NEXT OBJECT ADDR: D07F

ADDR DESCRIPTION/CONTENTS

D07F Wait for Drive to  
D081 come up to speed <D385>  
D089 Is motor on yet? <D4DA>  
D08C No, then exit with error >>D0EA  
D08E Is command a "status" request?  
D090 Yes, then determine status >>D0FD  
D092 Is command a "read" request?  
D093 Yes, then continue on >>D098  
D095 Prepare data for write (preinbblize) <D5F0>  
D098 ---  
D09A Initialize "retry" count at 64 (D369)  
D09D ---  
D09F Read an address field - Good read? <D398>  
D0A2 Yes, then continue on >>D0BE  
D0A4 Decrement "retry" count - More to try? (D369)  
D0A7 Yes, then try again >>D09D  
D0A9 NO, just in case indicate "I/O Error"  
D0AB Decrement "recalibration" count - More to try? (D36A)  
D0AE NO, then exit with error >>D0EA  
D0B0 Get "current" track (D35A)  
D0B3 Preserve it  
D0B4 Double it and  
D0B5 add 16 to it for recalibration  
D0B7 Reinitialize Retry Count  
D0BC Branch always taken >>D0CC  
D0C1 Was the right track found? (D35A)  
D0C4 Yes, then continue on >>D0D5  
D0C6 Get "current" track (D35A)  
D0C9 Preserve it  
D0CA Get track we found  
D0CB Double it  
D0CC Put new value in Device Track Table <D4D3>  
D0CF Get track we want  
D0D0 And go there <D10C>  
D0D3 Branch always taken >>D09D  
D0D8 Was the right sector found? (D357)  
D0DB NO, then try again >>D0A4  
D0DE Is command a "write" request?  
D0E0 Yes, then go do it >>D0F4  
D0E2 Read the data - Good read? <D3FD>  
D0E5 NO, then try again >>D0A4  
D0E7 Indicate no errors  
D0E9 BNE instruction, never taken  
D0EA Indicate error  
D0EB Preserve error number (D358)  
D0EE Get Slot  
D0F0 Turn motor off (C088)  
D0F3 Return to caller

Disk II Device Driver -- V1.2 -- 6 SEP 86 NEXT OBJECT ADDR: D0F3

ADDR DESCRIPTION/CONTENTS

D0F4 \*\*\*\*\* HANDLE WRITE REQUEST \*\*\*\*\*  
D0F4 Write data - Good write? <D500>  
D0F7 Yes, then exit >>D0E7  
D0F9 Indicate "write-protect error"  
D0FB Branch always taken >>D0EA  
D0FD \*\*\*\*\* GET STATUS \*\*\*\*\*  
D0FD Get Slot number  
D102 Check "write-protect" status (C08E)  
D105 Put result in Carry flag  
D106 Select read mode (C08C)  
D109 Exit with appropriate status >>D0F7  
D10C \*\*\*\*\* LOCATE DESIRED TRACK \*\*\*\*\*  
D10C Double the track number for proper phase  
D10D Preserve destination track \* 2 (D36F)  
D110 Turn all phases off <D125>  
D113 Get offset into Device Track Table <D4F1>  
D116 Get track (D359)  
D119 Update "current" track (D35A)  
D11C Get destination track (D36F)  
D11F Update Device Track Table (D359)  
D122 Move arm to desired track <D133>  
D125 Initialize phase number, starting with 3  
D127 ---  
D128 Clear a phase <D18A>  
D12B Decrement phase number - More to do?  
D12C Yes, then continue until all phases done >>D127  
D12E Divide track number by 2 (D35A)  
D132 Return to caller  
D133 \*\*\*\*\* ARM MOVE ROUTINE \*\*\*\*\*  
D133 Preserve track to find (D372)  
D136 Are we already there? (D35A)  
D139 Yes, then set appropriate phase and exit >>D187  
D13D Initialize phase count (halftracks) (D36B)  
D143 Preserve "current" track for comparisons (D371)  
D146 Subtract track to find to compute delta-tracks  
D147 Are we already there? (D372)  
D14A Yes, then clear prior phase and exit >>D183  
D14C Positive delta-tracks - go move arm out >>D155  
D14E Negative delta-tracks - Get absolute value delta-tracks less 1  
D150 Increment current phase to move in (D35A)  
D153 Branch always taken >>D15A  
D155 Compute absolute value delta-tracks less 1  
D157 Decrement current phase to move out (D35A)

Disk II Device Driver -- V1.2 -- 6 SEP 86      NEXT OBJECT ADDR: D15A  
 ADDR    DESCRIPTION/CONTENTS

Disk II Device Driver -- V1.2 -- 6 SEP 86      NEXT OBJECT ADDR: D15A  
 ADDR    DESCRIPTION/CONTENTS

D15A Compare delta-tracks with phases moved (D36B)  
 D15D Use smaller value for offset to delay tables >>D162  
 D162 Are we pointing at last table value Yet?  
 D164 Yes, then continue to use current offset >>D168  
 D166 Else, use new offset  
 D167 Set Carry flag for set phase operation  
 D168 Set a phase <D187>  
 D16B Get delay value from table (D373)  
 D16E Delay <D385>  
 D171 Get prior phase number (D371)  
 D174 Clear Carry flag for clear phase operation  
 D175 Clear a phase <D18A>  
 D178 Get delay value from table (D37C)  
 D17B Delay <D385>  
 D17E Increment phases moved (D36B)  
 D183 Delay <D385>  
 D187 Get "current" phase number (D35A)  
 D18A Use low two bits only, zero to three - 000000PP  
 D18C Multiply by two and bring in Carry - 000000PPC  
 D18D Merge in slot number - 0SSS0PPC  
 D18F Put in X-reg for following operation  
 D190 Toggle appropriate phase (C080)  
 D193 Restore slot number to X-reg  
 D195 Return to caller

D196 \*\*\*\*\* TABLE 1 \*\*\*\*\*  
 Read Translate Table with Prenibble  
 Bit mask Tables and Epilog Table in  
 unused areas

D196 Read Translate  
 Bit Mask 1  
 DIA0 00000000  
 DIA1 10000000  
 DIA2 01000000  
 DIA3 11000000

D1A4 Read Translate  
 Bit Mask 2  
 DIC0 00100000  
 DIC1 00100000  
 DIC2 00010000  
 DIC3 00110000

D1C4 Epilog Table (\$DE,\$AA,\$EB)

D200 \*\*\*\*\* TABLE 2 \*\*\*\*\*  
 Write Translate Table  
 Every 4th byte starting at \$D203

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

D200 Entry for Bit Mask 1  
 D201 Entry for Bit Mask 2  
 D202 Entry for Bit Mask 3  
 D203 Entry for Write Translate

D300 \*\*\*\*\* AUXILIARY BUFFER \*\*\*\*\*  
 Auxiliary Buffer (\$56 bytes) >>0056

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

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

D356 Track number  
 D357 Sector number  
 D358 Error number

Disk Device Track Table

D359 Table Entry  
 D359 Current Unit  
 D35A Current Track  
 D35B Slot 1, Devices 1 & 2  
 D35D Slot 2, Devices 1 & 2  
 D35F Slot 3, Devices 1 & 2  
 D361 Slot 4, Devices 1 & 2  
 D363 Slot 5, Devices 1 & 2  
 D365 Slot 6, Devices 1 & 2  
 D367 Slot 7, Devices 1 & 2

D359 Table Entry  
 D359 Current Unit  
 D35A Current Track  
 D35B Slot 1, Devices 1 & 2  
 D35D Slot 2, Devices 1 & 2  
 D35F Slot 3, Devices 1 & 2  
 D361 Slot 4, Devices 1 & 2  
 D363 Slot 5, Devices 1 & 2  
 D365 Slot 6, Devices 1 & 2  
 D367 Slot 7, Devices 1 & 2

D359 Table Entry  
 D359 Current Unit  
 D35A Current Track  
 D35B Slot 1, Devices 1 & 2  
 D35D Slot 2, Devices 1 & 2  
 D35F Slot 3, Devices 1 & 2  
 D361 Slot 4, Devices 1 & 2  
 D363 Slot 5, Devices 1 & 2  
 D365 Slot 6, Devices 1 & 2  
 D367 Slot 7, Devices 1 & 2

Disk II Device Driver -- V1.2 -- 6 SEP 86 NEXT OBJECT ADDR: D367

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

D369 Retry count (initially 64)  
 D36A Recalculation 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)  
 D36F Track found  
 D370 Checksum found  
 D370 Delay counter (high byte)  
 D371 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 deacceleration)

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 >>D3A5

D3A0 Increment count (high order byte) - Zero yet? (D36B)

D3A3 Yes, exit and indicate Read Error >>D3FB

D3A5 Read data register (C08C)

D3A8 Loop until data valid >>D3A5

D3AA Is it first address mark (\$D5)?

D3AC NO, then increment "must find" count >>D39D

D3AE Delay for data latch to clear

D3AF Read data register (C08C)

D3B2 Loop until data valid >>D3AF

D3B4 Is it second address mark (\$AA)?

D3B6 NO, then see if it's first address mark >>D3AA

D3B8 Initialize count for four byte read

D3BA Read data register (C08C)

D3BD Loop until data valid >>D3BA

D3BF Is it third address mark (\$96)?

D3C1 NO, then see if it's first address mark >>D3AA

D3C3 Set Interrupt flag

Disk II Device Driver -- V1.2 -- 6 SEP 86 NEXT OBJECT ADDR: D3C4

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

D3C4 Initialize checksum  
 D3C9 Read "odd" encoded byte IX1X1X1X (C08C)  
 D3CE Align "odd" bits XI1X1X1X1  
 D3CF Save for later (D36B)  
 D3D2 Read "even" encoded byte IX1X1X1X (C08C)  
 D3D7 Combine bytes XXXXXXXX (D36B)  
 D3DA Preserve data (Volume,Track,Sector,Checksum) (D36D)  
 D3DD Do checksum computation (D36C)  
 D3E0 Decrement counter - Finished field yet?  
 D3E1 No, do some more >>D3C6  
 D3E3 Is checksum computation zero?  
 D3E4 NO, then exit with carry set >>D3FB  
 D3E6 Read data register (C08C)  
 D3E9 Loop until data valid >>D3E6  
 D3EB Is it first trailing byte (\$DE)?  
 D3ED NO, then exit with carry set >>D3FB  
 D3EF Delay for data latch to clear  
 D3F0 Read data register (C08C)  
 D3F3 Loop until data valid >>D3F0  
 D3F5 Is it second trailing byte (\$AA)?  
 D3F7 NO, then exit with carry set >>D3FB  
 D3F9 Clear the Carry flag (no error)  
 D3FA Return to caller  
 D3FB Set the Carry flag (error occurred)  
 D3FC Return to caller

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

D3FD Convert slot number to an

D3FE absolute reference (i.e. \$60 -> \$EC)

D400 Modify code for current slot number (D45A)

D403 (i.e. \$C08C,X -> \$C0EC) (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 (D497)

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 \$20

D433 Decrement count - More to do?

D434 NO, then exit >>D46D

D436 Read data register (C08C)

D439 Loop until data valid >>D436

D43B Is is 1st header mark (\$D5)?

D43D NO, then try again >>D433

D43F Delay for register to clear

D440 Read data register (C08C)

Disk II Device Driver -- V1.2 -- 6 SEP 86 NEXT OBJECT ADDR: D443

ADDR	DESCRIPTION/CONTENTS
D443	Loop until data valid >>D440
D445	Is is 2nd header mark (\$AA)?
D447	No, then see if it is 1st header mark >>D43B
D449	Delay for register to clear
D44A	Read data register (C08C)
D44D	Loop until data valid >>D44A
D44F	Is is 3rd header mark (\$AD)?
D451	No, then see if it is 1st header mark >>D43B
D453	Initialize offset into data buffer
D457	Initialize checksum
D459	Read a data byte (C0EC)
D45E	Translate it (D100)
D461	Store it in Auxiliary buffer (D256)
D464	Compute running checksum
D466	Increment offset - More to do?
D467	Yes, then continue >>D457
D469	Reinitialize offset into data buffer
D46B	Branch always taken >>D472
D46D	Set carry flag indicating error
D46E	Return to caller
D46F	Store byte in Primary buffer (bottom third) (1000)
D472	Read a data byte (C0EC)
D477	Translate it and merge in (D100)
D47A	bits from Auxiliary buffer (D256)
D480	Increment offset - done yet?
D481	No, then do another >>D46F
D483	Save last byte for later, no time now
D484	Strip off last two bits XXXXXX00
D486	Reinitialize offset
D488	Read a byte (C0EC)
D48D	Translate it and merge in (D100)
D490	bits from Auxiliary buffer (D256)
D496	Store byte in Primary buffer (middle third) (1000)
D499	Increment offset - done yet?
D49A	No, then do another >>D488
D49C	Read a byte (C0EC)
D4A1	Strip off last two bits XXXXXX00
D4A3	Reinitialize offset
D4A5	Translate byte and merge in (D100)
D4A8	bits from Auxiliary buffer (D254)
D4AE	Store byte in Primary buffer (top third) (1000)
D4B1	Read a byte (C0EC)
D4B6	Increment offset - done yet?
D4B7	No, then do another >>D4A5
D4B9	Strip off last two bits XXXXXX00
D4BB	Is checksum valid? (D100)
D4BE	No, then exit with error >>D4CC
D4C0	Get slot number
D4C2	Read data register (C08C)
D4C5	Loop until data valid >>D4C2
D4C7	Is it 1st trailing mark (\$DE)?

Disk II Device Driver -- V1.2 -- 6 SEP 86 NEXT OBJECT ADDR: D4CA

ADDR	DESCRIPTION/CONTENTS
D4CA	Yes, then continue with carry clear >>D4CD
D4CC	Set Carry flag indicating error
D4CD	Get byte we stored away, we have time now
D4CE	Set proper offset
D4D0	Store byte in Primary buffer (offset \$55)
D4D2	Return to caller
D4D3	***** UPDATE DEVICE TRACK TABLE *****
D4D3	Get offset into Device Track Table <D4F1>
D4D6	Update Device Track Table (D359)
D4D9	Return to caller
D4DA	***** DETERMINE IF DRIVE IS ON (DATA CHANGING)*****
D4DA	Get slot number
D4DC	Initialize counter
D4DE	Read data register (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 DSS0000
D4F2	Get Unit number 0000DSSS
D4F4	Divide by 16 0000DSSS
D4F8	Put Drive into Carry 0000DSSS D
D4FA	Strip out Drive 0000DSSS D
D4FC	Roll left 0000DSSD
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 >>D5DF
D50C	Put transition byte from secondary buffer (D300)
D50F	Put 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

Disk II Device Driver -- V1.2 -- 6 SEP 86 NEXT OBJECT ADDR: D51E

ADDR DESCRIPTION/CONTENTS

```

D51E ---
D520 Write "sync" byte <D5E7>
D523 Decrement counter, done yet?
D524 NO, then do another >>D51E
D526 Write first data mark ($D5)
D52B Write second data mark ($AA)
D530 Write third data mark ($AD)
D535 Initialize checksum
D536 Initialize index into Auxiliary buffer
D538 Branch always taken >>D53D
D53A Get data byte (Auxiliary buffer) (D300)
D53D Exclusive-or with previous data byte (D2FF)
D540 Put result in X-reg for table lookup
D541 Lookup "disk byte" in table (D203)
D544 Get slot
D546 Write "disk byte" (C08D)
D54C Decrement index - Done with Auxiliary buffer?
D54D NO, then another byte >>D53A
D54F Get last byte of Auxiliary buffer
D551 Initialize index into Primary buffer
D553 Exclusive-or with next data byte (1000)
D556 Strip out last two bits XXXXXX00
D558 Put result in X-reg for table lookup
D559 Lookup "disk byte" in table (D203)
D55C Get slot
D55E Write "disk byte" (C08D)
D564 Get data byte (Primary buffer) (1000)
D567 Increment offset, end of this page?
D568 NO, then continue on >>D553
D56A Did buffer start on page boundary?
D56C Yes, then go write checksum >>D5C0
D56E Did buffer start one past page boundary?
D570 Yes, then go write last byte >>D5B3
D572 Carry indicates odd or even buffer end
D573 Get transition byte
D575 Write it (C08D)
D57B Get second transition byte
D57D Delay 2 cycles for correct timing
D57E Increment offset, buffer end on odd byte?
D57F Yes, go see if we're done then >>D599
D581 Exclusive-or with next data byte (1100)
D584 Strip out last two bits XXXXXX00
D586 Put result in X-reg for table lookup
D587 Lookup "disk byte" in table (D203)
D58A Get slot
D58C Write "disk byte" (C08D)
D592 Get data byte (Primary buffer - page 2) (1100)
D595 Increment offset
D596 Exclusive-or with next data byte (1100)
D599 End of buffer? - Put result in carry
D59B Strip out last two bits XXXXXX00

```

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

NEXT OBJECT ADDR: D59D

ADDR DESCRIPTION/CONTENTS

```

D59D Put result in X-reg for table lookup
D59E Lookup "disk byte" in table (D203)
D5A1 Get slot
D5A3 Write "disk byte" (C08D)
D5A9 Get data byte (Primary buffer - page 2) (1100)
D5AC Increment offset - Done yet?
D5AD NO, then do another >>D581
D5AF Yes, then go write checksum >>D5B1
D5B1 --- >>D5C0
D5B3 Get last byte
D5B6 Write it (C08D)
D5BC Delay 14 cycles for correct timing
D5C0 Use last byte in Primary buffer as checksum
D5C2 Lookup "disk byte" (D203)
D5C5 Get slot
D5C7 Write "disk byte" (C08D)
D5CD Initialize offset into "epilog" table
D5CF Delay 11 cycles for correct timing
D5D3 Load "epilog" from table ($DE,$AA,$EB,$FF) (D1C4)
D5D6 GO write it <D5E9>
D5D9 Increment offset
D5DA Done all four yet?
D5DC NO, then do another >>D5D3
D5DE Clear Carry flag (no error)
D5DF Select read mode (C08E)
D5E5 Return to caller

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

D5F0 ***** PRENIBLIZE BLOCK ROUTINE *****
D5F0 Get buffer pointer
D5F5 Add $2 to buffer address
D5F7 TO access top third of buffer >>D5FA
D5FA Store result in code below (D630)
D601 Subtract $54 from buffer address
D603 TO access middle third of buffer >>D606
D606 Store result in code below (D625)
D60D Subtract $AA from buffer address
D60F TO access bottom third of buffer >>D612
D612 Store result in code below (D61B)
D618 Initialize offset
D61A Get data byte (bottom third) XXXXXXXX (1000)
D61D Get last two bits 000000AB
D61F Put in X-reg for table lookup

```

Disk II Device Driver -- V1.2 -- 6 SEP 86      NEXT OBJECT ADDR: D620  
 ADDR    DESCRIPTION/CONTENTS

D620 Use lookup to reposition bits    0000BA00 (D1E0)  
 D623 Save result on stack  
 D624 Get data byte (middle third)    XXXXXXXX (1056)  
 D627 Get last two bits            000000CD  
 D629 Put in X-reg for table lookup  
 D635 Get current value from stack    0000BA00  
 D62B Merge in new bits using table    00DCBA00 (D1C0)  
 D62E Save result on stack            XXXXXXXX (10AC)  
 D632 Get last two bits            000000EF  
 D634 Put in X-reg for table lookup    00DCBA00  
 D635 Get current value from stack    00DCBA00  
 D636 Merge in new bits using table    FEDCBA00 (D1A0)  
 D639 Save result on stack  
 D63A Get offset into primary buffer  
 D63B Compute offset into Auxiliary buffer  
 D63D Put in X-reg  
 D63E Get data byte just created      FEDCBA00  
 D63F Store it in Auxiliary buffer (D300)  
 D642 Increment offset primary buffer, done yet?  
 D643 No, then do another >>D61A  
 D645 Get low order byte of buffer  
 D647 Subtract 1 (offset to last byte in buffer)  
 D648 Save it for later  
 D64A Get low order byte of buffer  
 D64C Modify code in Write Data Routine (offset) (D552)  
 D64F Buffer on page boundary? - Yes, skip ahead >>D65F  
 D651 Else, compute offset to last byte  
 D653 Before page boundary  
 D654 Get byte (page boundary -1)  
 D656 Point at next byte (page boundary)  
 D657 Exclusive-or them together      XXXXXXXX  
 D659 Strip off last two bits            XXXXXXX0  
 D65B Put in X-reg for table lookup  
 D65C Get "disk byte" from table (transition byte) (D203)  
 D65F Save result (0 indicates page boundary)  
 D661 Buffer on page boundary? - Yes skip ahead >>D66F  
 D663 Get offset to last byte in buffer  
 D665 Carry indicates odd or even buffer start  
 D666 Get byte (page boundary)  
 D668 Did buffer start on odd byte? - Yes skip >>D66D  
 D66A Point at next byte (page boundary +1)  
 D66B Exclusive-or them together  
 D66D Save result  
 D66F Point at last byte in buffer  
 D671 Get last byte in buffer            XXXXXXXX  
 D673 Strip off last two bits            XXXXXXX0  
 D675 Save result ("checksum byte")  
 D677 Get high order byte of buffer  
 D679 Modify code in Write Data Routine (D555)  
 D68C Get slot number for this operation

Disk II Device Driver -- V1.2 -- 6 SEP 86      NEXT OBJECT ADDR: D68E  
 ADDR    DESCRIPTION/CONTENTS

D68E Modify code in Write Data Routine (D55D)  
 D69A Return to caller  
 D69B \*\*\*\*\* DETERMINE IF SLOT/DRIVE HAS CHANGED \*\*\*\*\*  
 D69B Compare unit number with "current" unit number (D359)  
 D69E Put "current" drive in Carry  
 D69F Has slot changed? - No, then exit >>D6BD  
 D6A9 Get "current" slot  
 D6AB Put in X-register  
 D6AC Exit if Slot 0 >>D6BD  
 D6AE Is "current" motor is on? <D4DC>  
 D6B1 NO, then exit >>D6BD  
 D6B8 Wait until "current" motor is off (D370)  
 D6BB Or else timeout >>D6A6  
 D6BD Return to caller  
 D6BE \*\*\*\*\* CLEAR IWM PHASES \*\*\*\*\*  
 D6BE Get unit number  
 D6C0 Strip drive bit  
 D6C2 Put slot\*16 in X-Register  
 D6C3 Clear phases in case there is (C080)  
 D6C6 one of them new-fangled storage (C082)  
 D6C9 devices sharing this slot (C084)  
 D6CC with my (t)rusty old Disk II. (C086)  
 D6CF Return to caller  
 D6D0 \*\*\*\*\* CHECK CALLING PARAMETERS \*\*\*\*\*  
 Note: For 40-track drives, change byte at \$D6E3  
       from \$18 to \$40  
 D6D0 Check command code  
 D6D2 Is it greater or equal to 4?  
 D6D4 Yes, indicate error >>D6E6  
 D6D6 Get Block Number  
 D6DA Is Block Number good? (D356)  
 D6DD Yes, if less than \$100 >>D6E8  
 D6E0 No, if greater than or equal to \$200 >>D6E6  
 D6E4 No, if greater than or equal to \$118 >>D6E8  
 D6E6 Indicate error  
 D6E7 Return to caller  
 D6E8 All is well  
 D6E9 Return to caller  
 D6EA \*\*\*\*\* \$D6EA-\$D6FF NOT USED \*\*\*\*\*  
 D6EA Not used

```

Disk II Device Driver -- V1.3 -- 2 DEC 86      NEXT OBJECT ADDR: D6BE
-----
ADDR  DESCRIPTION/CONTENTS
-----

```

```

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

```

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

```

D6BE ***** CLEAR IWM PHASES *****

```

```

D6BE Get unit number
D6C0 Strip drive bit
D6C2 Put slot*16 in X-Register
D6C3 Turn off 8 phases
D6CF Return to caller

```

```

D6D0 ***** CHECK CALLING PARAMETERS *****
      Note: For 40-track drives, change byte at $D6E3
            from $18 to $40.

```

```

D6D0 Check command code
D6D2 Is it greater or equal to 4?
D6D4 Yes, indicate error >>D6E6
D6D6 Get Block Number
D6DA Is Block Number good? (D356)
D6DD Yes, if less than $100 >>D6E8
D6E0 NO, if greater than or equal to $200 >>D6E6
D6E4 NO, if greater than or equal to $118 >>D6E8
D6E6 Indicate error
D6E7 Return to caller
D6E8 All is well
D6E9 Return to caller

```

```

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

```

```

D6EA Not used

```



IRQ Handler -- V1.2 -- 6 SEP 86 NEXT OBJECT ADDR: FF9B

-----  
 ADDR DESCRIPTION/CONTENTS

FF9B MODULE STARTING ADDRESS  
 \*\*\*\*\*  
 \* IRQ Handler  
 \* Resides at \$FF9B. Put  
 \* there by ProDOS Relocator.  
 \*  
 \* VERSION 1.2 -- 6 SEP 86  
 \* VERSION 1.3 -- 2 DEC 86  
 \* (The IRQ Handler is still the  
 \* same as it was in Version 1.0.1)  
 \*  
 \* \*\*\*\*\*

FF9B \*\*\*\*\* GLOBAL PAGE EQUATES \*\*\*\*\*  
 BF56 Temporary storage 1  
 BF57 Temporary storage 2  
 BF88 A register savearea  
 BF8D Bank ID byte  
 BFD3 IRQ exit code

FF9B \*\*\*\*\* EXTERNAL EQUATES \*\*\*\*\*  
 D000 RAM/ROM test byte  
 C082 ROM Select  
 C08B BANK1 Select

FF9B \*\*\*\*\* IRQ CODE \*\*\*\*\*  
 FF9B Put A-Register on stack  
 FF9C Get Accumulator value from \$45  
 FF9E and save it (BF56)  
 FFAl Replace \$45 with A-Register  
 FFA2 since it may have been destroyed  
 FFA4 Load Status register  
 FFA5 Restore onto stack  
 FFA6 Isolate B flag - Was it a BRK?  
 FFA8 Yes, skip interrupt stuff >>FFC2  
 FFAA Else, Check location \$D000 (D000)  
 FFAD Do we have RAM active  
 FFAF Yes, indicate so >>FFB3  
 FFB1 Else, indicate ROM  
 FFB3 Update Bank ID byte (BF8D)  
 FFB6 Also save temporarily (BF57)  
 FFB9 Push (\$BF50) address of  
 FFBb routine to bank in Ram and  
 FFBc call IRQ on the stack  
 FFBF Push a new P-Register on stack with  
 FFC1 the Interrupt Disable flag set  
 FFC2 Push (\$FA41) address less 1 of

IRQ Handler -- V1.2 -- 6 SEP 86 NEXT OBJECT ADDR: FFC4

-----  
 ADDR DESCRIPTION/CONTENTS

FFC4 Monitor IRQ on the stack  
 FFC8 Select ROM - execution continues in ROM (C082)  
 \*\*\*\*\* RESET CODE \*\*\*\*\*  
 FFCB Push (\$FA61) address less 1 of (FFD7)  
 FFCE Hardware Reset routine on to stack  
 FFD3 Exit via select ROM code above >>FFC8  
 FFD6 Address (-1) of Hardware Reset routine  
 \*\*\*\*\* IRQ CODE \*\*\*\*\*  
 Called via \$BF50 in System Global Page

FFD8 Save Accumulator in Global page (BF88)  
 FFD8 Restore \$45 with original value (BF56)  
 FFE0 Select RAM (read & write) (C08B)  
 FFE3 use BANK1 (C08B)  
 FFE6 Get Bank ID byte (BF57)  
 FFE9 Leave via Global Page IRQ exit code >>BFD3

FFEC \*\*\*\*\* \$FFEC-\$FFF9 UNUSED \*\*\*\*\*  
 FFEC These unused bytes are at \$4FEC-\$4FF9 when  
 FFF9 loaded as part of the "PRODOS" file.  
 FFFA \*\*\*\*\* VECTORS \*\*\*\*\*  
 FFFA NMI Vector  
 FFFC Reset Vector  
 FFFE IRQ Vector

ThunderClock Code -- V1.2 -- 6 SEP 86 NEXT OBJECT ADDR: D742

ADDR 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 (1-31)  
 003D Binary hour of the day (0-23)  
 003E Binary minute of the hour (0-59)  
 D742 \*\*\*\*\* EXTERNAL EQUATES \*\*\*\*\*  
 0200 Input Buffer  
 BF90 Global page year-month-day  
 BF92 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 (0538)  
 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) <C10B>  
 D74E Read the clock. <C108>  
 Reading the clock results in an ASCII string being placed in the input buffer. A sample string might be "07,06,04,22,46,57", which is July (month 07) Saturday (day-of-week 6) the 4th (day of month 4) 10 PM (hour 22) 46 minutes and 57 seconds after the hour.

ThunderClock Code -- V1.2 -- 6 SEP 86 NEXT OBJECT ADDR: D751

ADDR DESCRIPTION/CONTENTS

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

D770 \*\*\*\*\* NOW CONVERT TO PRODOS DATE, TIME \*\*\*\*\*  
 D770 Save month in Y-reg  
 D771 three low bits of month  
 D774 three high bits of accum,  
 D775 combine with day of month,  
 D777 and store in low byte of DATE (MMDDDDDD) (BF90)  
 D77A Save carry (high bit of month)  
 D77B Add day of the month  
 D77D to table value to get Julian date (D7AB)  
 D780 Late September? >>D784  
 D782 Yes, add 3 and carry (see notes below)  
 D784 Compute Julian date MOD 7  
 D785 ---  
 D78B Subtract day-of-week to get year index  
 D78D Index positive? >>D791  
 D78F No, make it positive  
 D791 Index to Y-Reg  
 D792 Get year from year table (D7B8)  
 D795 Get high bit of month in carry  
 D796 roll it into accumulator  
 D797 to get high byte of DATE (YYYYYYM) (BF91)  
 D79A Put hour  
 D79C in high byte of TIME (BF93)  
 D79F Put minutes  
 D7A1 in low byte of TIME (BF92)  
 D7AB Restore saved screen hole value (0538)  
 D7AB RETURN

D7AC \*\*\*\*\* JULIAN TABLE \*\*\*\*\*  
 D7AC January  
 D7AD February  
 D7AE March  
 D7AF April  
 D7B0 May

ThunderClock Code -- V1.2 -- 6 SEP 86                   NEXT OBJECT ADDR: D7B1

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

D7B1   June  
D7B2   July

      If month>7, value in table is one  
      less than Julian, because a carry  
      is added along with the day of the month

D7B3   August  
D7B4   September

      Note: For Julian dates>255, the table value  
      is three more than the low byte should be,  
      so that it will properly divide by 7.

D7B5   October  
D7B6   November  
D7B7   December

D7B8   \*\*\*\*\* YEAR TABLE \*\*\*\*\*  
      This table is good for the years 1986-91

D7B8   1990  
D7B9   1989  
D7BA   1988 (March to December)  
D7BB   1988 (January and February)  
D7BC   1987  
D7BD   1986  
D7BE   1991

IIGS Clock Code -- V1.2 -- 6 SEP 86      NEXT OBJECT ADDR: D742  
 -----  
 ADDR    DESCRIPTION/CONTENTS

IIGS Clock Code -- V1.2 -- 6 SEP 86      NEXT OBJECT ADDR: D788  
 -----  
 ADDR    DESCRIPTION/CONTENTS

D742    MODULE STARTING ADDRESS  
 \*\*\*\*\*  
 \* IIGS CLOCK CODE \*\*\*\*\*  
 \* If ProDOS 8 is booted on  
 \* an Apple IIGS, then this  
 \* code is loaded into the  
 \* ProDOS data area at \$D742.  
 \*  
 \* VERSION 1.2 -- 6 SEP 86, and  
 \* VERSION 1.3 -- 2 DEC 86  
 \*  
 \*\*\*\*\*

D788    Roll high bit of month into year to put (BF91)  
 YYYYYYM in Global Page.  
 D78B    Throw 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.

D742 \*\*\*\*\* GLOBAL PAGE VALUES \*\*\*\*\*  
 BF90    ProDOS DATE word, YYYYYYMMDDDD.  
 BF92    ProDOS TIME word, 000HHHH00MMMM.

D742 \*\*\*\*\* IIGS SOFT SWITCH \*\*\*\*\*  
 C068    One byte sets 8 soft switches.

D742 \*\*\*\*\* CLOCK CODE ENTRY POINT \*\*\*\*\*  
 D742    8-bit memory and index operations.  
 D744    Get IIGS STATEREG Status Byte, (C068)  
 D747    save it temporarily. (D790)  
 D74A    Make sure we're in bank 0.  
 D750    Get out of emulation mode.  
 D751    16-bit memory and index operations.  
 D756    Push 4 null words on the stack.  
 D75A    Tool = ReadTimeHex. (0D03)  
 D75D    Jump to the tool dispatcher. <E10000>  
 D761    8-bit memory operations.  
 D763    Get pre-call Status Byte (D790)  
 D766    and restore all soft switches. (C068)  
 D769    Throw away seconds.  
 D76B    Store minutes in ProDOS Global Page. (BF92)  
 D76F    Store hours in ProDOS Global Page. (BF93)  
 D772    Get year  
 D773    and store it temporarily. (BF91)  
 D776    Get day of month  
 D777    in range 1-31  
 D778    and save it temporarily. (BF90)  
 D77B    Get month  
 D77C    in range 1-12  
 D77D    and shift it left five bits.  
 D782    Combine with the date (BF90)  
 D785    and store in Global Page as MMMDDDD. (BF90)

## HOW "BASIC.SYSTEM" IS LOADED AND RELOCATED

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

```

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

```

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

```

I-----I
I                                     I
I "BASIC.SYSTEM" I
I  21 BLOCK FILE I
I                                     I
I(20 data blocks I
I plus one index I--->
I block)         I
I                                     I
I    L$2800      I
I                                     I
I-----I
I                                     I
I-----I$4800
I  BI GLOBAL PAGE  I
I-----I$4700
I                                     I
I          BASIC          I
I                                     I
I    INTERPRETER         I
I                                     I
I    (load location)     I
I                                     I
I-----I$2400
I  BI RELOCATOR      I
I-----I$2000
I                                     I

```

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

BI Relocator -- V1.1 -- 18 JUN 84 NEXT OBJECT ADDR: 2000

ADDR DESCRIPTION/CONTENTS

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 \$9A00-\$BCFF.

\* BASIC VERSION 1.1 -- 18 JUN 84

\* DISTRIBUTED WITH PRODOS 8 VERSIONS

\* 1.1.1, 1.2, and 1.3.

\*\*\*\*\*

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

0000 "FROM" POINTER FOR COPY

0001

0002

0003 "TO" POINTER FOR COPY

0036 KSWL VECTOR

0038 KSWL VECTOR

006F APPLESOFT START OF STRINGS

0073 APPLESOFT HIMEM

00F2 APPLESOFT TRACE FLAG

\*\*\*\*\* EXTERNAL ADDRESSES \*\*\*\*\*

0200 PATHNAME BUFFER

0280 PREFIX BUFFER

0281 START OF PREFIX NAME

03D0 WARMSTART VECTOR

03D3 COLDSTART VECTOR

03F0 BRK HANDLER ADDRESS

03F1

03F2 RESET HANDLER ADDRESS

03F3

03F4 POWER-UP BYTE

03F5 APPLESOFT & VECTOR

03F8 CTL-Y VECTOR

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

BI Relocator -- V1.1 -- 18 JUN 84 NEXT OBJECT ADDR: 2000

ADDR DESCRIPTION/CONTENTS

0400 FIRST SCREEN BUFFER LINE

0480 SCREEN BUFFER LINE

0628 SCREEN BUFFER LINE

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

B7A BASIC INTERPRETER VERSION NUMBER

BE00 BASIC INTERPRETER ENTRY POINT

BE03 BI COMMAND SCANNER (SYNTAX)

BE10 COUT VECTORS FOR EACH SLOT

BE20 KSWL VECTORS FOR EACH SLOT

BE3C DEFAULT SLOT NO.

BE3D DEFAULT DRIVE NO.

BEFB HIMEM

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

BF00 MACHINE LANGUAGE INTERFACE ENTRY

BF30 LAST DEVICE USED

BF58 MEMORY MAP

BF98 MACHINE TYPE FLAGS

BF99 SLOTS WHICH CONTAINS CARDS WITH ROM

BF9A IF 0, NO PREFIX ACTIVE

BFFD INTERPRETER VERSION NUMBER

\*\*\*\*\* ROM ADDRESSES \*\*\*\*\*

E000 APPLESOFT ENTRY POINT

FA59 BRK HANDLER

FB2F INIT SCREEN, MONITOR, ETC.

FC58 CLEAR SCREEN, HOME CURSOR

FDED STANDARD CHARACTER OUT

FDF0 CHARACTER OUTPUT TO SCREEN

FEB4 SET NORMAL CHARACTER ATTRIBUTE

\*\*\*\*\* 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

204B \$02 --> \$9A00

2055 COPY 35 PAGES

2058 COPY INTERP TO HIGH MEMORY AT \$9A00 <20C4>

205D PAGE FOLLOWING INTERP IMAGE IS...

205F BASIC GLOBAL PAGE IMAGE

2061 COPY THAT TO \$BE00 <20C4>

2064 TO GET 40-COL DISPLAY, SEND A CTRL-U

BI Relocator -- V1.1 -- 18 JUN 84

NEXT OBJECT ADDR: 2066

ADDR DESCRIPTION/CONTENTS

```

2066 OUT THE NORMAL OUTPUT VECTOR. <FDED>
2069 SET NORMAL CHARACTER ATTRIBUTE <FE84>
206C INITIALIZE SCREEN/WINDOW <FE2F>
206F CLEAR SCREEN/HOME CURSOR <FC58>
2076 SET BITMAP TO MARK LOWER 48K FREE (BF58)
207C EXCEPT PAGES 0 AND 1 AND
207E TEXT PAGES 4 THROUGH 7 (BF58)
2086 MARK $9000-$BFFF IN USE..
2091 EXCEPT FOR $BA00-$BDFE ARE FREE
2096 LOOK AT LANGUAGE IN ROM (E000)
2099 IS IT APPLESOFT?
209B NO, THEN CAN'T RUN INTERP >>20B1
20A0 GOT AT LEAST 64K?
20A2 NO, THIS ONLY WORKS IN 64K >>20B1
20A6 SET MY CSWL/KSWL FOR INTERP INIT (221A)
20AC COPY ALL 4 BYTES >>20A6
20AE THEN GO TO BASIC COLDSTART >>E000
      (WE WILL GET CONTROL AT $20D4 AGAIN)

```

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

20B1 ---

```

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

```

20C4 \*\*\*\*\* COPY PAGES (\$0/1--&gt;\$2/3) \*\*\*\*\*

```

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

```

20D4 \*\*\*\*\* CSWL INTERCEPT / CONTINUE \*\*\*\*\*

```

20D4 "]" APPLESOFT PROMPT?
20D6 NO...DON'T PRINT WHATEVER IT IS >>20D3
20D8 YES, APPLESOFT DONE SETTING UP (BE10)
20DB POINT CSWL TO STANDARD OUTPUT
20E2 CHECK LAST DEVICE USED (BF30)
20E5 SET ONLINE PARAMETER TO THIS (2238)
20EB DRIVE ONE OR TWO? >>20EE
20EE STORE DEFAULT DRIVE (D) (BE3D)
20F2 ISOLATE SLOT FROM DEVICE NO.
20F7 AND STORE DEFAULT SLOT (S) (BE3C)
20FE GET SLOT BYTE SHOWING CARDS PRESENT (BF99)
2102 PICK OFF ITS BITS ONE BY ONE
2108 SET OUTVECS AND INVECS TO $CS00 (BE10)
210B FOR ALL SLOTS WITH ROMS IN THEM (BE20)

```

BI Relocator -- V1.1 -- 18 JUN 84

NEXT OBJECT ADDR: 2115

ADDR DESCRIPTION/CONTENTS

```

2115 ---
211B SET HIMEM TO $9600
211D IN VARIOUS PLACES
2124 GOT A DEFAULT PREFIX? (BF9A)
2127 NO >>214E
2129 YES, MLI: GET PREFIX <BF00>
212F ERROR? >>218B
2136 BACKSCAN 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
2148 NO, MLI: SET PREFIX <BF00>
214E MLI: ONLINE <BF00>
2154 ERROR? >>218B
2156 GET VOL NAME LENGTH (0281)
215B NONE THERE? >>218B
215F ADD ONE TO NAME LENGTH (0280)
2164 AND PREFIX IT WITH A "/" (0281)
2167 MLI: SET PREFIX <BF00>
216D ERROR? >>218B

```

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

```

216F MLI: GET FILE INFO <BF00>
2172 FIND "STARTUP" FILE
2175 ERROR? >>218B
217A SAVE LENGTH OF STARTUP FILE NAME (2236)
217D COPY NAME TO $200 (2006)
2186 FIRST COMMAND WILL BE "-STARTUP"
218B CHECK NUMBER OF SUBLEVELS (223E)
2190 MORE THAN JUST VOL? >>2198
2192 MLI: SET PREFIX <BF00>
2198 ANY STARTUP FILE NAME? (2236)
219B YES, SKIP MESSAGE >>21C1
219D SET TRUE KSWL <2209>
21A2 PRINT ' PRODOS BASIC 1.1' (2267)
21AD PRINT ' COPYRIGHT ... (2283)
21B6 SKIP THREE LINES

```

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

```

21C1 ---
21C3 COPY WARMSTART JMP TO PAGE 3 (21FF)
21C9 AND COLDSTART (03D3)
21CC AND CTL-Y (03F8)
21CF POINT & VECTOR (2206)
21D2 TO $BE03 (CMD SCANNER) (03F5)
21D8 COPY BRK HANDLER JMP ALSO (2202)
21E7 AND RESET JMP (03F2)
21F2 SET POWER-UP BYTE ACCORDINGLY (03F4)

```

BI Relocator -- V1.1 -- 18 JUN 84      NEXT OBJECT ADDR: 21F7  
 ADDR    DESCRIPTION/CONTENTS

BI Relocator -- V1.1 -- 18 JUN 84      NEXT OBJECT ADDR: 2283  
 ADDR    DESCRIPTION/CONTENTS

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

\*\*\*\*\* VECTOR ADDRESSES \*\*\*\*\*

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

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

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

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

221A CSWL (20D4) INTERCEPT ADDR  
 221C KSWL (2209) INTERCEPT ADDR

221E GET FILE INFO PARMLIST  
 221F FILE NAME IS AT \$2006  
 2221 15 BYTES RESERVED FOR OTHER GET\_FILE PARMS (NOT USED)  
 2230 THIS BYTE NOT USED

2231 SET PREFIX PARM LIST  
 2232 FOR PREFIX AT \$2234

2234 NULL PREFIX  
 2235 "/"

2236 SAVED LENGTH OF STARTUP FILE NAME

2237 ONLINE PARM LIST  
 2239 PUT VOLUME NAME AT \$281

223B SET PREFIX PARMLIST  
 223C PREFIX IS AT \$280

223E NUMBER OF SUBLEVELS IN PREFIX +1

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

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

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

2400 BASIC INTERP IMAGE



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.SYSTEM.
* IT PERFORMS COMMAND HANDLING
* FOR ALL BUILT-IN PRODOS COM-
* MANDS AND SUPPORTS BASIC'S FILE
* HANDLING.
*
* VERSION 1.1 -- 18 JUN 84
*
* DISTRIBUTED WITH PRODOS VERSIONS
* 1.1.1, 1.2, AND 1.3.
*
*****
***** ZERO PAGE ADDRESSES *****

```

```

0024 CURSOR HORIZONTAL
0028 SCREEN LINE BASE ADDR
0029
0033 MONITOR PROMPT CHARACTER
0036 CRT DISPLAY VECTOR (CSWL)
0037
0038 KEYBOARD INPUT VECTOR (KSWL)
0039
003A SCRATCH POINTER AND LOOP COUNTER
003B
003C SCRATCH POINTER AND LOOP COUNTER
003D
003E POINTER TO APPLESOFT VARIABLES
003F
0050 APPLESOFT: LINE NUMBER
0051
0067 APPLESOFT: START OF PROGRAM PTR
0068
0069 APPLESOFT: LOMEM (START OF VARS)
006A
006B APPLESOFT: START OF ARRAY VARS PTR
006C
006E APPLESOFT: START OF FREEAREA PTR
006D
006F APPLESOFT: START OF STRINGS PTR
0070
0073 APPLESOFT: HIMEM (END OF STRINGS)
0074
0075 APPLESOFT: CURRENT LINE BEING EXECUTED
0076

```

BASIC Interpreter (BI) -- V1.1 --18 JUN 84 NEXT OBJECT ADDR: 9A00

ADDR DESCRIPTION/CONTENTS

```

009B APPLESOFT: ADDR OF LINE AFTER FINDLINE
009C
00AF APPLESOFT: END OF PROGRAM PTR
00B0
00B8 APPLESOFT: START OF PROGRAM PTR
00B9
00D6 APPLESOFT: PROGRAM LOCKED (PROTECTED)
00D8 APPLESOFT: ONERR ACTIVE FLAG
00DE APPLESOFT: ONERR CODE
00F2 APPLESOFT: TRACE ACTIVE FLAG
00F8 APPLESOFT: INTERNAL STACK

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

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

```

```

BE06 EXTERNAL COMMAND ENTRY TO BI
BE0C PRINT ERROR MESSAGE ENTRY TO BI
BE0F PRODOS ERROR CODE
BE10 OUTPUT VECTORS FOR ALL SLOTS
BE30 CURRENT OUTPUT VECTOR
BE32 CURRENT INPUT VECTOR
BE34 PRODOS INTERCEPT VECTORS (INPUT/OUTPUT)
BE38 BI'S INTERNAL REDIRECTION VECTORS
BE3C DEFAULT SLOT
BE3D
BE3E A REGISTER SAVE AREA
BE3F X REGISTER SAVE AREA
BE40 Y REGISTER SAVE AREA
BE41 TRACE FLAG (APPLESOFT TRACE ON/OFF)
BE42 EXEC FILE ACTIVE=$80, DEFERRED=1
BE43 IMMEDIATE COMMANDS=0, DEFERRED=1
BE44 EXEC FILE ACTIVE=$80
BE45 WRITE FILE ACTIVE=$80
BE46 READING PREFIX ACTIVE=$80
BE47 DIRECTORY FILE BEING ACCESSED
BE49 FREE STRING SPACE DURING GARBAGE COLLECT
BE4A BUFFERED I/O BYTE COUNT
BE4B INDEX INTO INPUT COMMAND LINE
BE4C LAST OUTPUT CHAR TO PREVENT RECURSION
BE4D NUMBER OF OPEN NON-EXEC FILES
BE4E EXEC FILE BEING CLOSED FLAG
BE4F READ FILE IS TRANSLATED DIRECTORY
BE50 VECTOR TO EXTERNAL COMMAND HANDLER
BE52 LENGTH-1 OF EXTERNAL COMMAND STRING
BE53 COMMAND NUMBER
BE54 PARAMETERS ALLOWED FOR THIS COMMAND

```

BASIC Interpreter (BI) -- V1.1 --18 JUN 84 NEXT OBJECT ADDR: 9A00

ADDR DESCRIPTION/CONTENTS

(SEE BIT DEFINITIONS IN TABLE LATER)

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

BE56 A KEYWORD VALUE  
 BE58 B KEYWORD VALUE  
 BE5A E KEYWORD VALUE  
 BE5D L KEYWORD VALUE  
 BE5F S KEYWORD VALUE  
 BE61 D KEYWORD VALUE  
 BE62 F KEYWORD VALUE  
 BE63 R KEYWORD VALUE  
 BE65 @ KEYWORD VALUE  
 BE6A T KEYWORD VALUE  
 BE6B SLOT NUMBER FROM IN# OR PR#  
 BE70 ISSUE MLI CALL AND XLATE ERROR CODES  
 MLI PARM LIST FIELDS  
 BEA3 CREATE: ACCESS CODE  
 BEA4 CREATE: FILE ID  
 BEA5 CREATE: AUX ID  
 BEA7 CREATE: FILE KIND  
 BEB4 SET/GET FILE INFO: PARM COUNT  
 BEB7 SET/GET FILE INFO: ACCESS CODE  
 BEB8 SET/GET FILE INFO: FILE ID  
 BEB9 SET/GET FILE INFO: AUX ID  
 BEBB SET/GET FILE INFO: FILE KIND  
 BEBC SET/GET FILE INFO: BLOCKS USED  
 BEBE SET/GET FILE INFO: MODIFY DATE/TIME  
 BEC7 ONLINE/GET/SET MARK/EOF/BUF: REF NUM  
 BEC8 ONLINE/GET/SET MARK/EOF/BUF: MARK/BUF  
 BECE OPEN: SYSTEM BUFFER  
 BED0 OPEN: REF NUM RETURNED  
 BED2 NEWLINE: REF NUM  
 BED3 NEWLINE: NEW LINE CHAR (ALWAYS CR)  
 BED6 READ/WRITE: REF NUM  
 BED7 READ/WRITE: DATA ADDRESS  
 BED9 READ/WRITE: LENGTH OF DATA  
 BEDB READ/WRITE: ACTUAL LENGTH TRANSMITTED  
 BEDE CLOSE/FLUSH: REF NUM  
 BEFB BASIC HIMEM VALUE

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

BE03 QUIT VECTOR  
 BE30 LAST DEVICE USED  
 BF58 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: 9A00

ADDR DESCRIPTION/CONTENTS

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

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

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

D43F APPLESOFT RESTART ENTRY  
 D61A FIND LINE BY NUMBER IN APPLESOFT  
 D665 SET POINTERS IN APPLESOFT  
 D7D2 EXECUTE NEW APPLESOFT STATEMENT  
 D820 APPLESOFT CMD EXECUTE  
 D865 APPLESOFT SIGNAL ERROR  
 ED24 APPLESOFT PRINT DECIMAL NUMBER  
 F273 APPLESOFT SET NORMAL CHARS

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

FC58 MONITOR CLEAR SCREEN/HOME CURSOR  
 FC9C MONITOR CLEAR TO EOL  
 FD10 MONITOR READ KEY (NO CURSOR)  
 FDED COUT VECTOR

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

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

9A00 ---  
 9A01 REPLACE CSWL/KSWL WITH CURRENT (BE30)  
 9A04 ACTUAL DEVICE DRIVER VECTORS  
 9A16 RETURN

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

9A17 SET IMMEDIATE COMMAND MODE  
 9A19 AND GO SET I/O VECTORS <9F76>  
 9A1C KSWL/H ALREADY SET?  
 9A21 NO? THEN CHECK CSWL >>9A26  
 9A23 YES, CONTINUE >>9AA3  
 9A26 CSWL/H ALREADY SET?  
 9A2B YES, CONTINUE >>9AA3  
 9A2D NO, SAVE CURRENT INTERCEPTS FIRST >>9A8D

BASIC Interpreter (BI) -- V1.1 --18 JUN 84 NEXT OBJECT ADDR: 9AA2  
 ADDR DESCRIPTION/CONTENTS

```

9AA3 ***** SET CSWL/KSWL INTERCEPTS *****
-----
9AA3 ***** SET CSWL/KSWL INTERCEPTS *****
(IMMEDIATE MODE)
-----
9AA3 ***** INPUT INTERCEPT: MODE = 0 *****
9AA3 ***** IMMEDIATE MODE *****
-----
9ABA IS EXEC FILE ACTIVE? (BE43)
9ABD NO >>9AC5
9ABF YES, SAVE REGISTERS <9F62>
9AC2 AND GO READ EXEC FILE FOR INPUT COMMANDS >>9BAF
-----
9AC5 NO EXEC FILE, RESTORE REAL CSWL/KSWL <9A00>
9AC8 NO, READ A KEY FROM KEYBOARD <9A00>
9ACB RETURN?
9ACD NO, EXIT >>9AEB
9ACF YES, SAVE REGISTERS <9F62>
9AD2 STORE IT IN LINE BUFFER (0200)
--> THIS ENTRY CALLED BY EXEC TO PROCESS
A COMMAND STRING STORED AT $200
9AD5 GO PROCESS THE COMMAND STRING <A677>
9AD8 CHECK COMMAND NUMBER RETURNED FROM PARSE (BE53)
9ADB EXIT BI RIGHT NOW? >>9AEB
9ADD NO, COMMAND RETURNED WITH ERROR CODE? >>9AF0
9ADF NO, RESTORE Y REG (BE40)
9AE2 RETURN A BACKSPACE TO CALLER OF KEYBOARD
9AE4 AND A LINE INDEX OF ZERO
9AE6 EXIT THE BI >>9AEB
-----
9AE8 RESTORE CALLER'S REGISTERS <9F6C>
9AEB AND EXIT BI BY INSTALLING INTERCEPTS >>9A8D
-----
9AEE ***** ERROR HANDLER *****
-----
9AEE ERROR=3, "NO DEVICE CONNECTED"
9AF0 MAIN ENTRY: STORE ERROR CODE (BE0F)
9AF3 AND IN APPLESOFT ONERR
9AF5 CHECK BI STATE (BE42)
9AF8 MEMORIZE WHETHER IT'S IMMEDIATE MODE
9AFD SET A HIGH FILE LEVEL FOR NON-EXEC FILES (BF94)
9B02 NO ACTIVE READ/WRITE FILES OR PREFIX READ (BE44)
9B0B CLOSE ALL OPEN FILES AT OR ABOVE (BEDE)
9B0E 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 NEXT OBJECT ADDR: 9A2F  
 ADDR DESCRIPTION/CONTENTS

```

9A2F ***** OUTPUT INTERCEPT: MODE = 0 *****
(IMMEDIATE MODE)
-----
9A2F ***** CHARACTER? (9F61)
9A32 NO.. >>9A54
9A34 ELSE, SAVE X REG (BE3F)
9A38 CHECK STACK FOR $D812 AS RETURN ADDR (0103)
9A3B (APPLESOFT TRACE, PRINTING #LINENO)
9A44 NOT TRACING? >>9A6E
9A46 ELSE, SET DEFERED MODE=4
9A4B GET SET TO PRINT THE "#" (9F61)
9A4E RESTORE X REG (BE3F)
9A51 AND GO TO OTHER OUTPUT HANDLER >>B7F1
-----
9A54 NOT A #, SAME AS LAST OUTPUT THO? (BE4C)
9A57 (SAVE FOR NEXT TIME THRU) (BE4C)
9A5A NO, ALL IS WELL >>9A74
9A5C TWO RETURNS IN A ROW?
9A5E NO, ALL IS WELL >>9A74
9A60 HAS HORIZONTAL CURSOR POSN CHANGED?
9A62 YES.. >>9A69
9A64 ELSE, ANYTHING IN PATHNAME BUFFER? (BCBD)
9A67 (MUST BE ALPHA)
9A69 RESTORE A REG
9A6B PATHNAME IS THERE... >>9A74
9A6D ELSE, WE ARE RECURSING INFINITELY, EXIT!
9A6E WE WERE'NT TRACING AFTER ALL, RESTORE X (BE3F)
9A71 AND A REGS, THEN FALL THRU TO EXIT (9F61)
-----
9A74 ***** ECHO OUTPUT CHAR AND EXIT *****
-----
9A74 PUT BACK REAL CSWL/KSWL VECTORS <9A00>
9A77 OUTPUT THE CHARACTER <FDED>
9A7A WAS IT A RETURN?
9A7C NO, EXIT NOW >>9A8D
9A7E ELSE, WAS APPLESOFT TRACING?
9A82 YES >>9A8B
9A84 NO, CLEAR MY TRACE FLAG (PSEUDO TRACE NOW) (BE41)
9A87 FORCE APPLESOFT TO TRACE FOR MY BENEFIT ONLY
9A8B RESTORE A REG AND FALL THRU TO EXIT BI
-----
9A8D ***** SAVE ACTUAL IN/OUT VECTORS *****
-----
9A8D ---
9A8E COPY KSWL/H TO VECIN
9A98 AND CSWL/H TO VECOUT
9A9A IN BI GLOBAL PAGE (BE31)
-----

```

BASIC Interpreter (BI) -- V1.1 --18 JUN 84      NEXT OBJECT ADDR: 9B18  
 -----  
 ADDR    DESCRIPTION/CONTENTS  
 -----

9B18 ERROR? >>9B27  
 9B1A PUT FILE LEVEL BACK TO ZERO  
 9B22 NOW FLUSH ALL OPEN FILES  
 9B24 MLI: FLUSH (ALL) <BE70>  
 9B27 ---  
 9B28 ASSUME MODE WILL BE 4 (DEFERRED)  
 9B2A MEMORIZE WHETHER BASIC ONERR ACTIVE  
 9B2C DEFERRED MODE CURRENTLY? >>9B30  
 9B2E NO, STILL IMMEDIATE MODE (MODE=0)  
 9B30 ---  
 9B31 SET MODE AS DEFINED ABOVE <9F76>  
 9B34 RESTORE BI'S CSWL/KSWL INTERCEPTS <9AA3>  
 9B37 GET ERROR CODE (BE0F)  
 9B3B BASIC ONERR ACTIVE? THEN GO HANDLE IT >>9B4D  
 9B3E NO, JUST PRINT ERROR MESSAGE <BE0C>  
 9B41 CLOSE EXEC FILE IF ONE IS OPEN <B2FB>  
 9B45 DEFERRED MODE? >>9B53  
 9B47 IMMED. MODE, PRINT RETURN AND... <9FAB>  
 9B4A WARMSTART APPLESOFT >>D43F  
  
 9B4D RESTORE STACK FOR BASIC  
 9B52 PASS ERROR CODE TO BASIC  
 9B53 ---  
 9B55 JUMP INTO APPLESOFT ERROR HANDLER >>D865

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

9B73 \*\*\*\*\* INPUT INTERCEPT: MODE=4 OR 8 \*\*\*\*\*  
 9B73 SAVE REGISTERS <9F62>  
 9B76 PREFIX INPUT ACTIVE? (BE46)  
 9B79 NO >>9B7E  
 9B7B YES, GO DO SPECIAL HANDLING >>9D67  
 9B7E ELSE, IS READ FILE ACTIVE? (BE44)  
 9B81 NO >>9B86  
 9B83 YES, GO DO SPECIAL HANDLING FOR THAT >>9C16  
 9B86 ELSE, IS EXEC FILE ACTIVE? (BE43)  
 9B89 NO >>9BAF  
 9B8B YES, GET PROMPT CHARACTER  
 9B8D IT BETTER NOT BE A "]"  
 9B8F IT IS, RETURN TO IMMEDIATE MODE >>9B58  
 9B91 ELSE, SET TRUE CSWL/KSWL <9A00>

BASIC Interpreter (BI) -- V1.1 --18 JUN 84      NEXT OBJECT ADDR: 9B94  
 -----  
 ADDR    DESCRIPTION/CONTENTS  
 -----

9B94 AND PASS CALLER'S AREG TO REMOVE CURSOR (BE3E)  
 9B97 RESTORE Y-REGISTER (B140)  
 9B9A REMOVE CURSOR AND GET A KEYPRESS <FD10>  
 9B9D BACKSPACE?  
 9B9F NO, EXIT BI >>9BAC  
 9BA1 YES, CHECK PROMPT  
 9BA3 IF ITS A ">".  
 9BA5 THEN EXIT WITH THE BACKSPACE >>9BAA  
 9BA8 ELSE, IF AT START OF LINE, REPROMPT >>9B94  
 9BAA MIDDLE OF LINE, RETURN A BACKSPACE  
 9BAC EXIT BI TO CALLER >>9A8D

9BAF \*\*\*\*\* READ EXEC FILE \*\*\*\*\*  
 9BAF REMOVE CURSOR FROM SCREEN  
 9BB1 CHECK PROMPT CHARACTER  
 9BB3 IF ITS A ">".  
 9BB5 DO THINGS DIFFERENTLY >>9BEF2  
 9BB7 CHECK KEYBOARD (C000)  
 9BBA NO KEY READY? >>9BCD  
 9BBC GOT A KEY, IS IT CONTROL-C?  
 9BBE NO, IGNORE IT >>9BCD  
 9BC0 YES, CLOSE EXEC FILE <B2FB>  
 9BC3 IMMEDIATE MODE? (BE42)  
 9BC6 NO >>9C01  
 9BC8 YES, CLEAR KEYBOARD STROBE (C010)  
 9BCB AND GO START NEW LINE >>9C01  
 9BCD SET UP FOR EXEC LINE READ <9D8A>  
 9BD0 READ A LINE TO \$200 <9C6C>  
 9BD3 ERROR? >>9BEFA  
 9BD5 SAVE REGISTERS <9F62>  
 9BD8 HOP INTO LOOP >>9BDE  
 9BDA ---  
 9BDB BACKSCANNING \$200 BUFFER (0200)  
 9BDE FORCING THE MSB ON  
 9BE6 RESTORE TRUE CSWL/KSWL <9A00>  
 9BE9 GO PROCESS COMMAND LINE <9AD5>  
 9BEC CHECK COMMAND NUMBER (BE53)  
 9BEF IMMEDIATE EXIT? IF NOT, GET NEXT LINE >>9BCD  
 9BF1 RETURN

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

9BF2 GET SET TO READ EXEC LINE <9D8A>  
 9BF5 READ SINGLE CHARACTER PER CALL <9C48>  
 9BF8 NO ERRORS, EXIT TO CALLER NOW >>9BF1

BASIC Interpreter (BI) -- V1.1 --18 JUN 84      NEXT OBJECT ADDR: 9BF8  
 ADDR    DESCRIPTION/CONTENTS

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

```

9BF8 CLOSE EXEC FILE <B245>
9BF9 WAS ERROR "END OF DATA"?
9BFF NO, REAL ERROR THEN >>9C13
9C01 ELSE, OK -- JUST STOP EXECING
9C03 GET CURSOR HORIZONTAL POSITION
9C05 IF IN MID LINE, PASS SCREEN CHAR BACK >>9C0E
9C07 ELSE, CHANGE PROMPT TO "]"
9C0B AND RETURN WITH A BACKSPACE
9C0D RETURN
9C0E GET SCREEN CHARACTER UNDER CURSOR
9C10 AND EXIT THRU KSWL TO GET REAL KEYPRESS >>0038
9C13 REAL ERROR, GO TO BI'S MAIN ERROR HANDLER >>9AF0

```

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

```

9C16 GET PROMPT
9C18 IF ITS A "]"...
9C1C THEN RESET TO IMMEDIATE MODE >>9B58
9C1F ELSE, REMOVE CURSOR FROM SCREEN (BE3E)
9C24 CHECK KEYBOARD (C000)
9C27 NO KEYPRESS? >>9C31
9C29 GOT A KEY, IS IT CONTROL-C?
9C2B NO, IGNORE IT >>9C31
9C2D CLEAR STROBE AND EXIT TO CALLER (C010)
9C30 RETURN

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

```

```

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

```

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

```

9C48 SAVE CURRENT READ/WRITE COUNT (BED9)
9C4B IN L KEYWORD VALUE (BE5F)
9C50 SET UP TO READ ONE BYTE (BED9)
9C55 MLI: READ <BE70>
9C58 ERROR? >>9C66
9C5A PUT COUNT BACK TO MAXIMUM AGAIN (BE5F)
9C60 GET FIRST CHARACTER ON $200 LINE (BED7)

```

BASIC Interpreter (BI) -- V1.1 --18 JUN 84      NEXT OBJECT ADDR: 9C63  
 ADDR    DESCRIPTION/CONTENTS

```

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 <BE70>
9C71 ERROR? >>9C66
9C73 GET LENGTH ACTUALLY TRANSMITTED (BEDB)
9C76 NOTHING? >>9C8E
9C79 GOT SOMETHING, FIND END OF DATA (BED7)
9C7D FETCH LAST BYTE OF LINE (01FF)
9C82 IS IT A RETURN CHARACTER?
9C84 NO, LEAVE LINE ALONE >>9C8E
9C86 YES, WAS L KEYWORD GIVEN? (BE57)
9C8B 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
9CA5 MLI: GET MARK <BE70>
9CA8 ERROR? >>9D1F
9CAA ARE WE AT BEGINNING OF THIS FILE? (BEC8)
9CB0 NO, CONTINUE >>9CDF
9CB2 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 (BCB8)
9CCD AND THE NUMBER OF ENTRIES PER BLOCK (BD00)

```

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

```

9CD0 GO FORMAT NAME OF DIRECTORY <B0B8>
9CD3 STORE THE LENGTH OF LINE AT $200
9CD8 PUT A RETURN CHAR AT END OF LINE
9CDU AND EXIT TO CALLER
9CDE RETURN

```

BASIC Interpreter (BI) -- V1.1 --18 JUN 84      NEXT OBJECT ADDR: 9CDE  
 -----  
 ADDR    DESCRIPTION/CONTENTS  
 -----

9CDF    GET CAT FLAG (BE4F)  
 9CE2    IF ZERO, GO PROCESS INDIVIDUAL ENTRIES >>9D22  
 9CE4    IF MINUS, GO DO SUMMARY LINE OR EXIT >>9CF9  
 9CE6    POSITIVE, ASSUME NULL LINE WANTED  
 9CE8    DROP CAT FLAG BY ONE (BE4F)  
 9CEB    IF ZERO, JUST GO PRINT A BLANK LINE >>9CD3

\*\*\*\*\* FORMAT TITLE LINE \*\*\*\*\*  
 9CED    ELSE, BLANK OUT \$200 AND <A66C>  
 9CF2    UNPACK "NAME TYPE BLOCKS ETC... <9FB0>  
 9CF5    LINE LENGTH IS 80  
 9CF7    GO RETURN IT TO CALLER >>9CD3

\*\*\*\*\* FORMAT SUMMARY LINE \*\*\*\*\*  
 9CF9    DO SUMMARY LINE?  
 9CFB    NO, JUST EXIT (ALL DONE) >>9D1C  
 9CFD    YES, DROP CAT FLAG SO EXIT NEXT TIME (BE4F)  
 9D02    CLEAR READ/WRITE COUNT (BED9)  
 9D0A    MLI: READ <BE70>  
 9D0D    FORMAT BLOCKS FREE AND INUSE SUMMARY LINE <B0E7>  
 9D11    GET REF NUM (BED6)  
 9D14    AND COPY TO GET/SET LIST (BEC7)  
 9D18    NO ERRORS, EXIT >>9CF5  
 9D1A    ERROR, JUMP TO BI ERROR EXIT >>9D1F  
 9D1C    "END OF DATA" ERROR  
 9D1F    GO TO BI ERROR EXIT >>9AF0

\*\*\*\*\* FORMAT FILE/DIR ENTRIES \*\*\*\*\*  
 9D22    SET DIR ENTRY NUM COUNTER TO -1  
 9D27    GET REF NUM (BED6)  
 9D2A    \*32  
 9D2F    USE AS INDEX TO GET ENTRY LENGTH (BCFF)  
 9D35    AND ENTRIES PER BLOCK FROM OPEN FILE LIST (BD00)  
 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 <BD1>  
 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 (BI) -- 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 = "]"?  
 9D69    NO, ALL IS WELL >>9D6E  
 9D6B    YES, RETURN TO IMMEDIATE MODE NOW >>9B58  
 9D6E    REMOVE CURSOR FROM SCREEN (BE3E)  
 9D75    PREFIX NO LONGER ACTIVE AFTER THIS (BE46)  
 9D7B    COPY PATHNAME BUFFER (PREFIX) (BCBC)  
 9D7E    TO \$200 (01FF)  
 9D84    RETURN WITH IT TO BASIC (BCBC)  
 9D89    RETURN

9D8A    \*\*\*\*\* SETUP TO READ LINE FROM EXEC \*\*\*\*\*  
 9D8A    SET READ REF NUM FOR EXEC FILE (BCA3)  
 9D90    READ TO \$200  
 9D95    FOR \$EF BYTES OF LENGTH  
 9D9A    (OR UNTIL A RETURN CHAR)  
 9DA2    RETURN

9DA3    \*\*\*\*\* OUTPUT INTERCEPT: MODE = C \*\*\*\*\*  
 (LOOK FOR CONTROL-D)

9DA3    SAVE REGISTERS <9F62>  
 9DA6    PRINTING A CONTROL-D?  
 9DA8    NO >>9DC1  
 9DAA    YES, WRITE OUT ANY BUFFERED DATA <9FF4>  
 9DAD    NOTHING IN COMMAND LINE (BE4B)  
 9DB0    READ FILE INACTIVE (BE44)  
 9DB3    WRITE FILE INACTIVE (BE45)  
 9DB6    PREFIX READ INACTIVE (BE46)  
 9DBB    SET MODE = 8 FROM NOW ON <9F76>  
 9DBE    RESTORE REGS AND EXIT >>9F6C

9DC1    GOT A CONTROL-D...  
 9DC3    SET MODE = 4 FROM NOW ON <9F76>  
 9DC6    RESTORE REGISTERS <9F6C>  
 9DC9    OUTPUT CHARACTER AND EXIT >>B7F1

9DCC    \*\*\*\*\* OUTPUT INTERCEPT: MODE = 8 \*\*\*\*\*  
 (ASSEMBLE COMMAND LINE)

9DCC    SAVE REGISTERS <9F62>  
 9DD2    SAVE CHAR IN COMMAND LINE (0200)  
 9DD5    WAS IT A RETURN?  
 9DD7    YES, READY TO ROLL >>9DE7  
 9DD9    NO, BUMP CHARACTER COUNTER (BE4B)

BASIC Interpreter (BI) -- V1.1 --18 JUN 84 NEXT OBJECT ADDR: 9DDC

BASIC Interpreter (BI) -- V1.1 --18 JUN 84 NEXT OBJECT ADDR: 9E50

ADDR DESCRIPTION/CONTENTS

ADDR DESCRIPTION/CONTENTS

9DDC AND EXIT TO CALLER >>9DE3  
 9DDE OOPS! LINE TOO LONG  
 9DE0 "SYNTAX ERROR" >>9AF0  
 9DE3 ELSE, RESTORE X REG AND EXIT (BE3F)  
 9DE6 RETURN  
 9DE7 ---  
 9DE9 NULL LINE? >>9DF6  
 9DEB NO, PUT BACK TRUE CSWL/KSWL <9A00>  
 9DEE SYNTAX SCAN CMD LINE <A677>  
 9DF1 ERROR? >>9DE0  
 9DF3 NO, PUT BACK BI'S INTERCEPTS <9A8D>  
 9DF6 ---  
 9DF8 MODE = 4 NOW <9F76>  
 9DFB RESTORE REGS AND EXIT >>9F6C

9DFE \*\*\*\*\* WRITE BUFFERED CHARACTER \*\*\*\*\*

9DFE SAVE Y REG (BE40)  
 9E01 CHECK PROMPT  
 9E03 CHECK TO SEE IF WE ARE IN "IF", >>9E11  
 9E06 "PRINT", "LIST", OR "CALL" STATEMENTS >>9E11  
 9E09 OF AN APPLESOFT PROGRAM >>9E11  
 9E0B IF NOT, EXIT TO CALLER... (BE40)  
 9E0E WITH CHARACTER ECHOED TO SCREEN >>9A74  
 9E11 GET INDEX TO TEMPORARILY BUFFERED CHARS (BE4A)  
 9E16 STORE INTO BUFFER JUST ABOVE HIMEM  
 9E1B BUMP INDEX (BE4A)  
 9E1E OK >>9E2B  
 9E20 BUFFER FULL, SAVE REGISTERS <9F62>  
 9E23 WRITE BUFFER OUT TO DISK <9FEE>  
 9E26 ERROR? >>9DE0  
 9E28 RESTORE REGISTERS <9F6C>  
 9E2B AND EXIT ANYWAY

9E2C \*\*\*\*\* OUTPUT INTERCEPT: MODE = 4 \*\*\*\*\*  
 (INITIAL ENTRY FOR A RUNNING PROGRAM)  
 (FLUSH OUT NON COMMAND LINES)

9E2C PRINTING A "#"? (9F61)  
 9E2F NO >>9E49  
 9E31 YES, SAVE X REGISTER (BE3F)  
 9E35 RETURN ADDR IS IN APPLESOFT... (0103)  
 9E38 TRACE ROUTINE...  
 9E3C AT \$D812? (0104)  
 9E41 YES >>9E86  
 9E43 NO, RESTORE REGISTERS (9F61)  
 9E49 IS WRITE FILE ACTIVE? (BE45)  
 9E4C NOPE >>9E6C  
 9E4E YES, PRINTING A "]"?

9E50 NO >>9E56  
 9E52 YES, SAME AS PROMPT CHARACTER?  
 9E54 YES >>9E86  
 9E56 NO, PRINTING A RETURN CHAR?  
 9E58 NO >>9DFE  
 9E5A YES, GET PROMPT  
 9E60 DOES IT INDICATE RECURSION? >>9DFE  
 9E62 YES, WRITE BUFFER OUT <9FF4>  
 9E65 OUTPUT FILE INACTIVE NOW (BE45)  
 9E6A EXIT WITH RETURN CHAR >>9E9F

---  
 9E6C INPUT FILE ACTIVE? (BE44)  
 9E73 NO >>9E7D  
 9E75 YES, CHECK PROMPT  
 9E77 OR IN \$04  
 9E79 CONTROL-D?  
 9E7B YES >>9EA2  
 9E7D ---  
 9E7E NO, HOW BOUT "]"?  
 9E80 NO, EXIT WITH ECHO THEN >>9E9F  
 9E82 YES, IS THIS THE PROMPT CHAR?  
 9E84 NO, EXIT WITH ECHO >>9E9F  
 9E86 YES, SAVE REGISTERS <9F62>  
 9E89 CHECK OPEN FILE COUNT (BE4D)  
 9E8C NONE OPEN? >>9E9C  
 9E8E SOME OPEN, WRITE BUFFER OUT <9FF4>  
 9E91 INDICATE WRITE FILE INACTIVE NOW (BE45)  
 9E94 SET TRUE CSWL/KSWL <9A00>  
 9E99 PRINT "FILE(S) STILL OPEN" <BE0C>  
 9E9C RESTORE REGS <9F6C>  
 9E9F AND ECHO EXIT >>9A74

---  
 9EA2 CHAR IS A RETURN?  
 9EA3 NO >>9EAA  
 9EA7 YES, SAME AS LAST CHAR OUTPUT? (BE4C)  
 9EAA (SAVE IT FOR THIS TEST NEXT TIME) (BE4C)  
 9EAD NOT SAME, NO PROBLEM THEN >>9EB1  
 9EAF SAME, MARK PROMPT FOR RECURSION  
 9EB1 RETURN

9EB2 \*\*\*\*\* APPLESOFT TRACE INTERCEPT \*\*\*\*\*  
 (CONTROL PASSES HERE FOR EVERY STATEMENT)  
 (EXECUTED WHILE PRODOS IS ACTIVE)

9EB2 BUMP APPLESOFT LINE POINTER  
 9EB6 ---  
 9EBA MARK PROMPT FOR RECURSION  
 9EBC JUST IN CASE WE DIE IN HERE  
 9EBE RESTORE APPLESOFT'S STACK

BASIC Interpreter (BI) -- V1.1 --18 JUN 84 NEXT OBJECT ADDR: 9EC1

ADDR DESCRIPTION/CONTENTS

```

9EC1 DOES BI KNOW WE ARE TRACING? (BE41)
9EC4 YES, REAL LIVE TRACE THEN >>9F39

9EC6 ELSE, PICK UP NEXT TOKEN ON LINE
9ECA IS IT A TOKEN? >>9EF1
9ECC OR END OF LINE? >>9EEE
9ECE NEITHER, DECREMENT STRING SPACE CTR (BE49)
9EDI OK >>9EEC
9ED3 COMPUTE SIZE OF FREESPACE IN PAGES
9ED7 AT LEAST 3 PAGES AVAILABLE?
9ED9 YES >>9EE5
9EDB NO, WRITE BUFFERED DATA <9FF4>
9EDE AND THEN GARBAGE COLLECT <A044>
9EE3 COMPUTE FREE SPACE NOW
9EE5 AND SAVE IN STRING SPACE CTR (BE49)
9EEA GET NEXT TOKEN
9EEC ---
9EEE JUMP BACK INTO APPLESOFT TO EXECUTE IT >>D820
9EF1 STORE TOKEN IN PROMPT
9EF4 LOOK UP TOKEN IN BI'S TOKEN TABLE (B799)
9EF7 IT'S NOT ONE BI IS INTERESTED IN >>9EEE
9EF9 IT IS INTERESTING, CHANGE BRANCH (9EF8)
9EFC AND JUMP TO ONE OF THE FOLLOWING: >>9EFE

9EFE IF OR PRINT: PROMPT = 0
9F00 CLEAR OUT LAST CHAR SAVEAREA (BE4C)
9F03 GO TO MODE = C NEXT TIME THRU (B803)
9F06 (BEGIN LOOKING FOR COMMANDS) (BE38)
9F0F NOW GO PROCESS THE IF OR PRINT >>9F2E

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

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

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

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

9F26 NOTRACE: DROP INTO BACKGROUND TRACE (BE41)
9F29 CHANGE TOKEN TO "TRACE"
9F2A FORCE ON APPLESOFT TRACE
9F2E ---
9F2F GO BACK TO APPLESOFT TO PERFORM IT >>D820

```

BASIC Interpreter (BI) -- V1.1 --18 JUN 84 NEXT OBJECT ADDR: 9F2F

ADDR DESCRIPTION/CONTENTS

```

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

***** REAL TRACE ACTIVE *****

9F39 RESTORE TRUE CSWL/KSWL <9A00>
9F3E PRINT "#" <FDED>
9F45 USE APPLESOFT TO PRINT CURRENT LINE NO. <ED24>
9F4A PRINT A BLANK SPACE <FDED>
9F4D PUT BI'S CSWL/KSWL INTERCEPTS BACK <9A8D>
9F51 THEN GO BACK AND HANDLE AS USUAL >>9EE6

9F54 LOOKING FOR A LOWER CASE "c"
9F58 LOOKING FOR A "#"
9F5A STORE CHAR TO SEARCH FOR (9FG1)
9F5E BRANCH BACK INTO APPLESOFT >>9EEC
9F60 BREAK IF Y IS ZERO!!!

9F61 "#" CHARACTER (ASOFT TRACE CHAR)

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

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

9F76 ***** SET MODE AND CSWL/KSWL *****
9F76 STORE "STATE" MODE FROM X REGISTER (BE42)
9F7B COPY PROPER CSWL/KSWL VALUES TO REDIRECT... (B7F7)
9F7E VECTOR DEPENDING ON CURRENT MODE (BE38)
9F87 RETURN

9F88 ***** PRINTERR: PRINT ERROR MSG *****
9F88 ---
9F89 GET INDEX INTO PACKED MESSAGE TEXTS (BA13)
9F8C UNPACK MESSAGE INTO $201 <9FB0>
9F92 SAVE THE LENGTH (BCB6)
9F95 SKIP A LINE <9FAB>
9F9A PRINT A BELL <9FAD>
9F9D ---
9F9F PRINT CONTENTS OF $201 MSG BUFFER (0201)
9FAB PRINT A RETURN CHARACTER
9FAD AND EXIT >>FDED

```



9FB0 \*\*\*\*\* UNPACK ERROR MESSAGE \*\*\*\*\*  
 BASIC Interpreter (BI) -- V1.1 --18 JUN 84 NEXT OBJECT ADDR: 9FAD  
 ADDR DESCRIPTION/CONTENTS

9FB0 \*\*\*\*\* UNPACK ERROR MESSAGE \*\*\*\*\*  
 9FB0 NOTHING IN BUFFER AT FIRST  
 9FB6 GET A NIBBLE FROM PACKED MSG <9FD2>  
 9FB9 NON-ZERO, COMMON CHARACTER >>9FC0  
 9FBB IF ZERO, GET NEXT NIBBLE <9FD2>  
 9FBE AND CONVERT TO UNCOMMON CHAR INDEX  
 9FC0 ---  
 9FC1 GET THE LETTER THIS NIBBLE REPRESENTS (BA28)  
 9FC4 ZERO? THEN END OF MESSAGE >>9FD1  
 9FC6 GET INDEX INTO OUTPUT BUFFER (BE4B)  
 9FC9 AND STORE THE CHARACTER THERE (0201)  
 9FCC BUMP INDEX (BE4B)  
 9FCF AND CONTINUE >>9FB6  
 9FD1 RETURN

9FD2 \*\*\*\*\* UNPACK MESSAGE BYTE \*\*\*\*\*  
 9FD2 GET NEXT MSG BYTE (BA48)  
 9FD5 WORKING ON SECOND NIBBLE? >>9FE9  
 9FD7 NO, TAB INDICATOR? >>9FDF  
 9FD9 NO, ISOLATE HIGH NIBBLE  
 9FDD NEXT TIME GET LOW NIBBLE  
 9FDE RETURN

9FDF ---  
 9FE0 GET TAB POSITION (BA48)  
 9FE3 AND BUMP OUTPUT PTR ACCORDINGLY (BE4B)  
 9FE7 THEN GO BACK FOR NEXT NIBBLE >>9FD2  
 9FE9 BUMP BYTE PTR FOR NEXT TIME  
 9FEA ISOLATE LOW NIBBLE  
 9FEC NEXT TIME GET HIGH NIBBLE  
 9FED RETURN

9FEE \*\*\*\*\* WRITE ONE BUFFERED BYTE \*\*\*\*\*  
 9FEE SET UP COUNT OF 0001  
 9FF2 AND JUMP INTO ROUTINE BELOW >>A007

9FF4 \*\*\*\*\* WRITE BUFFERED DATA/TEST ERROR \*\*\*\*\*  
 9FF4 WRITE BUFFERED DATA <A000>  
 9FF7 OK? THEN EXIT >>A01C  
 9FFA ERROR, POP OUT OF THIS SUBROUTINE  
 9FFD AND GO TO ERROR HANDLER >>9AF0

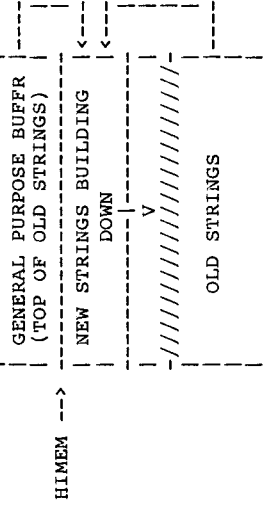
A000 \*\*\*\*\* WRITE ALL BUFFERED DATA \*\*\*\*\*  
 BASIC Interpreter (BI) -- V1.1 --18 JUN 84 NEXT OBJECT ADDR: 9FFD  
 ADDR DESCRIPTION/CONTENTS

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

A01D \*\*\*\*\* SPECIAL GARBAGE COLLECT \*\*\*\*\*  
 (PULL OUT STRING CONSTANTS ALSO)

A01D DO GARBAGE COLLECTION NORMALLY FIRST <A044>  
 A020 ERROR? >>A043  
 A024 START OF STRING AREA = PROGRAM START PTR (BC84)  
 A02C USE GENERAL PURPOSE BUFFER (ABOVE HIMEM)  
 A02E FOR A GARBAGE COLLECT WORKAREA (BC7D)  
 A033 IT IS 3+1 PAGES IN LENGTH (BC7E)  
 A038 END OF STRING AREA IS AT END OF FREEAREA (BC86)  
 A040 GO COLLECT CONSTANT STRINGS NOW <A085>  
 A043 THEN EXIT

A044 \*\*\*\*\* "FRE" COMMAND \*\*\*\*\*  
 (FAST APPLESOFT STRING GARBAGE COLLECTION)



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.

BASIC Interpreter (BI) -- V1.1 --18 JUN 84      NEXT OBJECT ADDR: A044  
 -----  
 ADDR    DESCRIPTION/CONTENTS  
 -----

A044 STRING AREA START IS ON PAGE BOUNDARY  
 A04B ASSUME 4 PAGE WORKAREA (BC7E)  
 A050 IN GENERAL PURPOSE BUFFER ABOVE HIMEM (BC7D)  
 A055 STRING START PTR IS START OF STRING AREA (BC84)  
 A059 COMPUTE NUMBER OF FREE PAGES  
 A05B AT LEAST 7?  
 A05D IF NOT, USE G.P. WORKAREA INSTEAD >>A079  
 A05F DON'T USE ALL OF FREE AREA (LEAVE \$300)  
 A061 NEW WORKAREA SIZE IS FREE AREA SIZE-\$300 (BC7E)  
 A066 SET PTR TO WORKAREA AT FIRST FREE PAGE  
 A06D COMPUTE NUMBER OF STRING PAGES  
 A071 USE SMALLER OF STRING PAGES OR WORKAREA SIZE (BC7E)  
 A076 AS NEW WORKAREA SIZE (BC7E)  
 A079 END OF STRING AREA IS HIMEM  
 A085 RECORD WHETHER LAST PAGE IS PARTIAL  
 A089 STRING START MSB IS HIMEM INITIALLY (BC86)  
 A08E ADJUST LORANGE AND HIRANGE MSB'S  
 A090 FOR PARTIAL PAGES AT EITHER END, (BC7F)  
 A093 SETTING THEM AT HIMEM FOR NOW.  
 A09C SET UP ARRAY END MSB +1 FOR COMPARES (BC82)  
 A09F \$3E/\$3F --> FIRST VARIABLE (LESS 7 BYTES)  
 A0A1 (EACH VARIABLE IS 7 BYTES)  
 A0AB SET UP ARRAY START LSB FOR COMPARES  
 A0B0 GET LORANGE VALUE (BC7F)  
 A0B3 PRIOR TO STRING AREA? (BC84)  
 A0B6 YES, THEN DONE! >>A0F6  
 A0B8 ELSE, DROP LORANGE BY WORKAREA SIZE (BC7E)  
 A0BB AND SAVE THIS VALUE (BC7C)  
 A0BE NOW DROP IT ALSO BY THE DISTANCE BETWEEN  
 A0C0 ..THE OLD LORANGE AND THE STRING START PTR (BC7F)  
 A0CA USE THE LOWER OF THE TWO VALUES (BC7C)  
 A0CF TO PRODUCE THE MAXIMUM SIZED RANGE (BC7C)  
 A0D2 IS THIS BELOW THE BOTTOM OF THE STRINGS? (BC84)  
 A0D5 NO >>A0DC  
 A0D7 YES, USE THE BOTTOM POINTER INSTEAD (BC84)  
 A0DA (ADJUSTING FOR PARTIAL PAGE)  
 A0DC STORE FINAL LORANGE VALUE (BC7F)  
 A0DF COPY SOME PAGES BELOW HIRANGE TO WORKAREA <A195>  
 A0E2 (TO MAKE ROOM FOR NEW STRINGS)  
 A0E4 COLLECT SIMPLE STRING VARS FOR THIS RANGE <A0F7>  
 A0E7 ERROR? >>A0F4  
 A0E9 THEN COLLECT STRING ARRAYS <A12D>  
 A0EC NEW HIRANGE = OLD LORANGE (BC7F)  
 A0F2 CONTINUE LOOPING >>A09F  
  
 A0F4 IF ERROR, "RAM TOO LARGE"  
 A0F6 EXIT TO CALLER

BASIC Interpreter (BI) -- V1.1 --18 JUN 84      NEXT OBJECT ADDR: A0F6  
 -----  
 ADDR    DESCRIPTION/CONTENTS  
 -----

A0F7 \*\*\*\*\* COLLECT SIMPLE STRINGS \*\*\*\*\*  
 A0F8 ---  
 A0F8 ADD 7 BYTES TO \$3E/\$3F PTR FOR NEXT VAR  
 A102 PTR AT ARRAYS NOW?  
 A108 IF SO, WE ARE DONE >>A12B  
 A10A IS THIS A STRING VARIABLE?  
 A111 NO >>A0F7  
 A113 MAKE ABSOLUTELY SURE  
 A117 GET MSB OF STRING POINTER  
 A11B IS IT WITHIN MY RANGE? (BC7F)  
 A123 NO >>A0F7  
 A125 YES, PULL IT OUT AND TACK IT TO HIMEM <A1B8>  
 A128 ALL WENT WELL, GET NEXT VARIABLE >>A0F8  
 A12A IF ERROR, EXIT NOW  
  
 A12B NORMAL EXIT TO CALLER  
 A12C RETURN  
  
 A12D \*\*\*\*\* COLLECT STRING ARRAYS \*\*\*\*\*  
 A12D FIND THE NEXT ARRAY <A15C>  
 A130 NO MORE? >>A12B  
 A132 GOT ONE, GET MSB OF ITS STRING PTR  
 A136 WITHIN MY RANGE? (BC7F)  
 A139 NO >>A146  
 A13E NO >>A146  
 A140 YES, PULL IT OUT AND TACK IT TO HIMEM <A1B8>  
 A143 AND CONTINUE WITH NEXT ARRAY ELEMENT >>A147  
 A145 ERROR EXIT  
  
 A146 ---  
 A147 BUMP POINTER TO NEXT ARRAY MEMBER  
 A151 POINTER NOW AT NEXT ARRAY? (BC81)  
 A154 NO, DO THIS ELEMENT >>A132  
 A158 NO >>A132  
 A15A YES, SET UP TO PROCESS THAT ONE THEN >>A12D  
  
 A15C \*\*\*\*\* FIND NEXT STRING ARRAY \*\*\*\*\*  
 A15C ---  
 A15D \$3E --> ARRAY VARIABLES (BC81)  
 A164 AT END OF ARRAY VARS  
 A166 NO, CONTINUE >>A16C  
 A16A YES, OUT (CARRY SET, NO MORE ARRAYS) >>A194

BASIC Interpreter (BI) -- V1.1 --18 JUN 84 NEXT OBJECT ADDR: A16A  
 -----  
 ADDR DESCRIPTION/CONTENTS  
 -----

A16C POINT TO ARRAY FOLLOWING THIS (LSB AND...)  
 A176 MSB TO X REGISTER  
 A17D CHECK TYPE OF VARIABLE  
 A182 SKIP INTEGER AND REAL ARRAYS >>A15C  
 A186 GET NUMBER OF DIMENSIONS  
 A188 \*2 TO SKIP SIZES  
 A189 +5 TO SKIP FIXED STUFF AT BEGINNING  
 A18D POINT TO FIRST ARRAY MEMBER  
 A191 READY TO ROLL, \$3E POINTS TO IT  
 A194 RETURN

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

A195 \$3A/\$3B --> FIRST PAGE TO SAVE (BC7C)  
 A19A \$3C/\$3D --> WORKAREA (BC7D)  
 A1A5 COPY N+1 PAGES (SIZE OF WORKAREA) (BC7E)  
 A1A9 ---  
 A1B7 EXIT WHEN FINISHED

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

A1B8 IS STRING BELOW SAVED AREA? (BC7C)  
 A1BB YES, ITS STILL THERE THEN >>A1C4  
 A1BD ELSE, POINT TO SAVED STRING IN WORKAREA (BC7C)  
 A1C4 \$3A/\$3B --> STRING  
 A1CF DROP STRING START PTR BY LEN OF THIS STRING  
 A1D4 UPDATE STRING'S LSB IN VARIABLE PTR  
 A1D8 FIX UP MSB OF STRING START PTR ALSO  
 A1DD AND OF VARIABLE PTR  
 A1E1 IS THIS A NULL LENGTH STRING?  
 A1E3 YES, NO MOVE TO DO >>A1EE  
 A1E6 ---  
 A1E7 ELSE, COPY STRING OUT  
 A1EE ---  
 A1EF OUT OF FREESPACE? (BC82)  
 A1F4 RETURN TO CALLER WITH INDICATION

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

BASIC Interpreter (BI) -- V1.1 --18 JUN 84 NEXT OBJECT ADDR: A1F5  
 -----  
 ADDR DESCRIPTION/CONTENTS  
 -----

\*\*\*\*\* 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, "RAM TOO LARGE" MSG  
 A208 TOO FEW... >>A24A  
 A20A GOT ENOUGH, \$3A->TOP OF FREESPACE  
 A211 AND \$3C-->NEW TOP AFTER ALLOCATION  
 A21B COMPUTE LENGTH OF STRINGS FOR COPY  
 A229 COPY STRINGS DOWN "N" PAGES IN MEMORY <A35B>  
 A22F SUBTRACT "N" FROM STRING ADDRESS MSB'S (BB47)  
 A235 ADJUST ALL POINTERS IN SIMPLE & ARRAY VARS <A39F>  
 A23A OLD HIMEM BECOMES BUFF ADDR HIGH WATER MARK (BB49)  
 A241 NEW HIMEM IS "N" PAGES LOWER  
 A246 FIND PAGE JUST BEYOND A FILE BUFFER (BC88)  
 A249 RETURN  
 A24A ---  
 A24B RETURN

A24C \*\*\*\*\* FREE BUFFER \*\*\*\*\*

A24C GARBAGE COLLECT STRINGS <A044>  
 A24E ERROR? >>A299  
 A255 PUT HIMEM-\$100 INTO \$3A/3B  
 A259 AND HIMEM-\$400 INTO \$3C/3D  
 A25F (COPY LSB'S)  
 A266 BC92 = LENGTH OF STRINGS (BC92)  
 A270 COPY STRINGS UP 4 PAGES <A37F>  
 A275 PREPARE TO ADJUST THEM BY \$400 (BC87)  
 A27B NEW HIMEM+\$400  
 A27D ADJUST ALL STRING ADDRS UP BY \$400 <A39F>  
 A283 ARE WE FREEING BOTTOM-MOST BUFFER?  
 A285 YES, DONE! >>A2B3  
 A288 CHECK OPEN FILE COUNT (BE4D)  
 A28B NONE OPEN? (HOW CAN THAT BE?) >>A297  
 A28D WHICH FILE'S BUFFER IS NEXT TO HIMEM?  
 A292 SEARCH UNTIL IT IS FOUND... >>A29A  
 A297 ---  
 A299 RETURN IF NO FILE IS USING THIS BUFFER  
 A29A ---  
 A29B GIVE THAT FILE THE BUFFER PASSED TO US (BEC9)  
 A29E (SURE HOPE THAT FILE WAS FLUSHED!) (BC93)  
 A2A9 PASS FILE REF NUM TO MLI (BEC7)  
 A2AE MLI: SET NEW BUFFER <BE70>  
 A2B1 ERROR? >>A299  
 A2B3 ---  
 A2B4 RETURN

BASIC Interpreter (BI) -- V1.1 --18 JUN 84 NEXT OBJECT ADDR: A2B4  
 ADDR DESCRIPTION/CONTENTS

A2B5 \*\*\*\*\* GETBUFR: GET A BUFFER \*\*\*\*\*  
 THIS ROUTINE IS CALLED THROUGH AN EXTERNAL  
 ENTRY POINT IN THE GLOBAL PAGE. IT ALLO-  
 CATES A FIXED LOCATION BUFFER BETWEEN THE  
 BI AND ITS BUFFERS.

A2B5 ALLOCATE A BUFFER OF ANY SIZE (A=PAGES) <A1F7>  
 A2B8 ERROR? >>A300  
 A2BD FIND FIRST PAGE OF BUFFER (BB4A)  
 A2C4 GET FILE OPEN COUNT (BE4D)  
 A2C7 NONE OPEN? >>A2EA  
 A2C9 BUMP BUFFER PAGE PTR BY \$400 (BB49)  
 A2CD TO POINT TO PREVIOUSLY ALLOCATED  
 A2CF BUFFER. (BB49)  
 A2D2 FIND OPEN FILE WITH THIS BUFFER (BC93)  
 A2D7 GOT IT. (BEC9)  
 A2DA SET FILE BUFFER REAL LOW IN MEMORY <A352>  
 A2DD THEN SET IT TO NEW BUFFER LOCATION <A29B>  
 A2E0 BELOW ALL OTHERS (BEC9)  
 A2E7 DO THIS FOR EACH OPEN FILE...  
 A2E8 THEREBY INSERTING A BLANK BUFFER >>A2D2  
 A2ED IS EXEC FILE ACTIVE? (BE43)  
 A2F0 NO, DONE >>A2FF  
 A2F2 YES,  
 A2F4 MOVE EXEC BUFFER DOWN ALSO <A352>  
 A2FD AND BUMP UP ABOVE IT  
 A2FF EXIT TO CALLER  
 A300 RETURN

A301 \*\*\*\*\* FREEBUFR: FREE BUFFER \*\*\*\*\*  
 THIS ROUTINE IS CALLED THROUGH AN EXTERNAL  
 ENTRY POINT IN THE GLOBAL PAGE. IT FREES  
 A FIXED LOCATION BUFFER PREVIOUSLY ALLO-  
 CATED BY GETBUFR.

A301 GET COUNT OF OPEN FILES (BE4D)  
 A305 INDEX THIS BY 4 PAGES PER FILE  
 A306 ADD TO HIMEM MSB  
 A308 SAVE THIS AS TOP OF BUFFERS (BB49)  
 A30D THEN SET UP BOTTOM AS HIMEM MSB (BB4A)  
 A310 GET OLD ORIGINAL HIMEM (BEFORE ANY BUFFERS) (BEFB)  
 A313 SAME AS THIS ONE?  
 A315 THEN NOTHING ELSE TO DO >>A350  
 A317 ASSUME NO BUFFERS BY REPLACING OLD HIMEM  
 A319 ANY EXEC FILE OPEN? (BE43)  
 A31C NO, CONTINUE >>A323  
 A31E YES, MOVE EXEC BUFFER TO OLD HIMEM <A2F2>  
 A321 AND GO MOVE HIMEM DOWN BY \$400 >>A341  
 A323 ELSE, START WITH TOP BUFFER (BB49)  
 A326 ANY OPEN FILES? (BE4D)

BASIC Interpreter (BI) -- V1.1 --18 JUN 84 NEXT OBJECT ADDR: A329  
 ADDR DESCRIPTION/CONTENTS

A329 IF NOT, WE ARE DONE >>A34D  
 A32B SEARCH FOR OPEN FILE WITH THIS BUFFER (BC93)  
 A32E NOT IT? >>A34A  
 A330 GOT IT, GIVE IT NEW HOME AT HIMEM  
 A332 AND SET BUFFER LOW <A352>  
 A335 THEN TO NEW LOC <A29B>  
 A339 DROP TOP BUFFER PTR BY \$400 (BB49)  
 A341 AND DROP HIMEM BY \$400  
 A348 AND GO DO NEXT BUFFER >>A323  
 A34A ---  
 A34B (LOOP TO SEARCH FOR OPEN FILES) >>A32B  
 A34D WHEN FINISHED, GARBAGE COLLECT <A044>  
 A350 ---  
 A351 THEN EXIT NORMALLY TO CALLER

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

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

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

A37F \*\*\*\*\* COPY BLOCK UP IN MEMORY \*\*\*\*\*  
 A37F PARTIAL PAGE? (BC92)  
 A382 NO, JUST COPY FULL PAGES NOW >>A38B  
 A384 YES, COPY SHORT PAGE FIRST <A396>  
 A387 DROP BOTH MSB'S  
 A38B PAGE COUNT GONE TO ZERO? (BC93)  
 A38E YES, DONE >>A39E  
 A390 ELSE, DROP PAGE COUNT (BC93)  
 A393 AND GO COPY A FULL PAGE UP >>A384

--- COPY REMAINDER OF PAGE UP (BACKWARDS)  
 A397  
 A39E RETURN

BASIC Interpreter (BI) -- V1.1 --18 JUN 84 NEXT OBJECT ADDR: A39E  
 ADDR DESCRIPTION/CONTENTS

BASIC Interpreter (BI) -- V1.1 --18 JUN 84 NEXT OBJECT ADDR: A40C  
 ADDR DESCRIPTION/CONTENTS

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

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

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

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

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

STRINGS
ARRAY VARS
SIMPLE VARS

A40D GARBAGE COLLECT FIRST <A01D>  
 A410 ERROR? >>A471  
 A412 COMPUTE LENGTH OF SIMPLE AND ARRAY VARS  
 A417 AND SAVE IT (BC89)  
 A427 NEXT, COMPUTE LENGTH OF SIMPLE VARS ONLY  
 A42B AND SAVE IT (BC8B)  
 A435 SUBTRACT VAR LENGTH FROM STRING START  
 A437 TO FIND A PLACE TO PUT THE VARS UNDER (BC92)  
 A43A THE STRINGS (START ON AN EVEN PAGE BOUND)  
 A440 \$3C/\$3D --> PLACE TO PUT VARS  
 A447 \$3A/\$3B --> START OF VARS (ROUNDED TO EVEN  
 A449 PAGE ALIGNMENT)  
 A44F COPY VARS UP AGAINST STRINGS <A37F>  
 A454 STORE START OF VARS PTR (BC8E)  
 A457 BUMPING PAGE NUMBER BY ONE  
 A463 SUBTRACT THIS PTR FROM HIMEM TO COMPUTE (BC90)  
 A466 TOTAL LENGTH OF COMBINED VARS/STRINGS  
 A468 AND SAVE THIS TOO (BC8D)  
 A46B ALSO, SAVE HIMEM MSB IN CASE THEY ARE MOVED  
 A471 DONE, EXIT

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

STRINGS
FREE SPACE

BASIC Interpreter (BI) -- V1.1 --18 JUN 84 NEXT OBJECT ADDR: A472

ADDR DESCRIPTION/CONTENTS

- A472 SAVE LENGTH OF SIMPLE AND ARRAY VARS (BC89)
- A479 ADD THIS TO START OF COMPRESSED VARS PTR
- A47B TO GIVE START OF STRINGS (\$6D/\$6E)
- A487 \$3C/\$3D --> LOMEM (WHERE TO PUT SIMPLE VARS)
- A48E \$6B/\$6C --> WHERE TO PUT ARRAY VARS
- A499 \$3A/\$3B --> START OF COMPRESSED VARS (BC8E)
- A4A3 COPY SIMPLE/ARRAY VARS DOWN TO LOMEM <A35B>
- A4AA COMPUTE START OF STRINGS BY ADDING VARS
- A4AC LENGTH TO VARS ORIGIN
- A4B5 DID HIMEM MOVE SINCE VARS WERE COMPRESSED?
- A4BA NO... >>A4C2
- A4BC YES, ADJUST BY DIFFERENCE IN HIMEM'S (BC87)
- A4BF GO ADJUST ALL STRING POINTERS <A39F>
- A4C2 THEN EXIT
- A4C3 RETURN

A4C4 \*\*\*\*\* FORHAT CATALOG ENTRY LINE \*\*\*\*\*

- A4C4 PUT OUT A BLANK LINE <A66C>
- A4C7 DOUBLE QUOTE TO S200
- A4CC GET LENGTH OF NAME (\$259)
- A4D2 COPY NAME TO LINE (\$259)
- A4DD ZERO ACCUMULATOR FOR LATER (BCB1)
- A4E0 GET FILE TYPE (\$269)
- A4E3 I KNOW OF ONLY 13
- A4E5 ---
- A4E7 LOOK UP FILE TYPE IN TABLE (B989)
- A4EA FOUND IT? >>A4F7
- A4F0 FILE TYPE NOT IN MY TABLE
- A4F2 PRINT IT IN HEXADECIMAL <A612>
- A4F5 AND CONTINUE BELOW >>A53B
- A4F7 ELSE, FOR KNOWN TYPES
- A4FA COPY NAME OF TYPE TO THE LINE (B997)
- A505 80 COLUMNS PER LINE? (BCB6)
- A508 YES... >>A553
- A50A NO,
- A50C BIN FILE?
- A50E YES... >>A525
- A510 TXT FILE?

BASIC Interpreter (BI) -- V1.1 --18 JUN 84 NEXT OBJECT ADDR: A512

ADDR DESCRIPTION/CONTENTS

- A512 NO... >>A53B
- A514 YES, R VALUE GIVEN AS SUBTYPE
- A51F CONVERT R VALUE TO DECIMAL <A62F>
- A522 SKIP OVER BIN CODE >>A536
- A525 BIN FILE, USE AD VALUE AS SUBTYPE
- A52D CONVERT IT TO TWO HEX DIGITS <A612>
- A536 ADD AN "=" SIGN
- A53B COPY MSB OF END OF FILE MARK (\$270)
- A549 CONVERT LOW TWO BYTES OF EOF <A62F>
- A550 DO CREATION DATE/TIME <A570>
- A553 ---
- A55B CONVERT BLOCKS USED <A62F>
- A563 CHECK FOR WRITE ACCESS
- A565 UNLOCKED? >>A56C
- A567 NO, ADD A ""
- A56C FALL THRU TO DO LAST MODIFIED DATE/TIME
- A56E AND THEN EXIT TO CALLER

A570 \*\*\*\*\* FORMAT A DATE/TIME \*\*\*\*\*

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

- A570 ISOLATE YEAR (\$25A)
- A574 AND STORE IT (BCB5)
- A57B ISOLATE DAY
- A57D AND STORE IT (BCB4)
- A581 ISOLATE MONTH
- A587 (MONTH = 0 IS NO GOOD) >>A5A3
- A58B (MONTH > 12 IS ALSO BAD) >>A5A3
- A58D STORE MONTH (BCB3)
- A591 MULTIPLY MONTH INDEX BY 3 (BCB3)
- A594 AND SAVE IT INSTEAD (BCB3)
- A59A (DAY = 0 IS NO GOOD) >>A5A3
- A5A1 (YEAR MUST BE < 99) >>A5B5
- A5A3 OTHERWISE, BAD DATE!
- A5A5 BACK UP 6 CHARACTERS ON LINE
- A5AA AND PRINT "<NO DATE>" (B9E5)
- A5B4 THEN EXIT RIGHT AWAY
- A5B5 DATE OK, GET HOUR (\$25C)
- A5B9 AND MINUTES (\$25B)
- A5BE MINUTES > 60?
- A5C0 NO... >>A5C3
- A5C2 YES, USE ZERO MINUTES
- A5C3 CONVERT MINUTES (LEFT ZERO FILL) <A60A>
- A5C8 THEN PRINT A ":" (\$201)
- A5CC GET HOUR AGAIN
- A5CF GREATER THAN 24 HOURS?
- A5D1 NOPE >>A5D4
- A5D3 YES, USE ZERO

BASIC Interpreter (BI) -- V1.1 --18 JUN 84 NEXT OBJECT ADDR: A5D4  
 ADDR DESCRIPTION/CONTENTS

A5D4 10 OR MORE HOURS (TWO DIGITS?)  
 A5D7 IN ANY CASE, CONVERT HOURS <A62F>  
 A5DB IF TWO DIGITS... >>A5DE  
 A5DD IF ONE, ADJUST LINE PTR  
 A5DE ---  
 A5E2 CONVERT YEAR (LEFT ZERO FILL) <A60A>  
 A5E6 GET MONTH INDEX (\*3) (BCB3)  
 A5E9 POINT TO LAST CHARACTER  
 A5EC COPY MONTH NAME FROM TABLE (B9BD)  
 A5EE TO LINE (0201)  
 A5F7 BACKWARDS... >>A5EC  
 A5FB PUT A "-" IN (0201)  
 A5FE TWO PLACES (0205)  
 A607 EXIT BY CONVERTING DAY >>A62F

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

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

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

A612 ---  
 A613 ISOLATE LOW NIBBLE  
 A615 AND GO CONVERT IT FIRST <A61D>  
 A619 NOW ISOLATE HIGH NIBBLE  
 A61C AND FALL THRU TO CONVERT IT ALSO  
 A61D CONVERT NIBBLE TO NUMERIC ASCII  
 A61F >9?  
 A621 NO >>A625  
 A623 YES, CONVERT \$BA-\$BF TO \$C1-\$C6  
 A625 AND STORE THE RESULT (0201)  
 A628 BUMP LINE INDEX BACK  
 A629 PRECEED WITH A \$ SIGN  
 A62E RETURN

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

A62F A,X = NUMBER Y=INDEX TO LAST FIELD DIGIT (BCB0)  
 A632 STORE NUMBER IN ACCUMULATOR (BCAF)  
 A635 DIVIDE BY 10 <A64D>  
 A638 GET DIGIT AND CONVERT IT (BCB2)  
 A63D STORE IN LINE (0201)  
 A640 AND DROP LINE INDEX BY ONE  
 A641 IS QUOTIENT NOW ZERO? (BCAF)  
 A64A NO, CONTINUE UNTIL IT IS >>A635

BASIC Interpreter (BI) -- V1.1 --18 JUN 84 NEXT OBJECT ADDR: A64C  
 ADDR DESCRIPTION/CONTENTS

A64C ELSE, EXIT  
 \*\*\*\*\* DIVIDE ACCUMULATOR BY 10 \*\*\*\*\*  
 A64D 24 BIT SHIFT (3 BYTES)  
 A651 CLEAR SUM (BCB2)  
 A654 GO ROL ACCUMULATOR LEFT ONE BIT <AAD7>  
 A657 ALSO ROL 4TH BYTE OF ACCUM (BCB2)  
 A65B IF MSB > 10... (BCB2)  
 A665 THEN ADD ONE TO ACCUMULATIVE SUM (BCAF)  
 A668 ---  
 A669 SHIFT 24 TIMES >>A654  
 A66B RETURN  
 A66C ---  
 A676 RETURN

A677 \*\*\*\*\* SYNTAX: PARSE COMMAND LINE \*\*\*\*\*  
 (ALSO EXTERNAL ENTRY FOR COMMAND STRINGS)

A677 INIT COMMAND NUMBER TO -1  
 A67E A BLANK ENDS EACH STRING (BCA9)  
 A683 AT MOST 8 CHARACTERS IN A COMMAND (BCAA)  
 A686 PARSE COMMAND ITSELF <AAIB>  
 A689 GET FIRST LETTER (BCBD)  
 A68C MUST BE ALPHABETIC  
 A68E IT IS... >>A697  
 A690 IT'S NOT, IS IT A "-"?  
 A692 YES, OK THEN... >>A697  
 A694 ELSE, ITS BAD - SYNTAX ERROR >>A839  
 A697 SCAN FOR COMMAND IN TABLES <AAEL>  
 A69A BAD COMMAND? >>A694  
 A69C NO, IMMEDIATE COMMAND MODE? (BE42)  
 A69F NO, DEFERRED... >>A6AC  
 A6A1 IMMEDIATE, EXEC ACTIVE? (BE43)  
 A6A4 YES, NEVER MIND >>A6AC  
 A6A6 ERASE TO END OF LINE <FC9C>  
 A6A9 AND GO TO A NEW LINE ON SCREEN <9FAB>  
 A6AC ASSUME NO PARMS AT ALL  
 A6B4 NO PATH NAME YET (BCBD)  
 A6B7 NO SECONDARY PATH NAME EITHER (0280)  
 A6BD CURRENT SLOT = DEFAULT SLOT (BE61)  
 A6C3 CURRENT DRIVE = DEFAULT DRIVE (BE62)  
 A6C8 BUFFER ALLOCATION = HIMEM (BC88)  
 A6D0 GET LENGTH OF COMMAND NAME (BE52)  
 A6D3 ALLOW 2 MORE CHARACTERS FOR NOW (BCAA)  
 A6D6 ARE ANY PARAMETERS PERMITTED? (BE54)  
 A6D8 NO...MUST BE MOM OR NOMON >>A736  
 A6D9 YES... >>A739  
 A6DB ELSE, REPARSE THE COMMAND <AALB>  
 A6E0 FOR THIS COMMAND... (BE54)

BASIC Interpreter (BI) -- V1.1 --18 JUN 84 NEXT OBJECT ADDR: A6E3

ADDR DESCRIPTION/CONTENTS

A6E3 DOES THE PREFIX NEED FETCHING? >>A6EA  
 A6E5 YES,  
 A6E7 MLI: GET PREFIX FROM DEFAULT DRIVE <BE70>  
 A6EA ---  
 A6EE END OF LINE? >>A736  
 A6EE NO, COMMA?  
 A6F0 NO >>A6F5  
 A6F2 YES, NO FILENAME, LOOK FOR KEYWORDS >>A787  
 A6F5 "/"?  
 A6F7 YES >>A6FD  
 A6F9 NO, ALPHABETIC?  
 A6FB NO...FILE NAMES MUST BEGIN THAT WAY >>A72F  
 A6FD ---  
 A6FE DON'T FLUSH ANY BLANKS OUT OF PATHNAME  
 A703 ALLOW 64 CHARACTERS NEXT PARSE  
 A709 PARSE NEXT OPERAND ON LINE <AA1F>  
 A70D SAVE ITS LENGTH (BCBC)  
 A712 FOUND A PATHNAME#1 (BE56)  
 A715 COPY PARM KEYWORD TO \$280 (BCBC)  
 A718 (ASSUMING PATHNAME1=PATHNAME2) (0280)  
 A71F CHECK NEXT CHAR (OTHER THAN A BLANK) <AA3A>  
 A722 NOT COMMA OR RETURN, BAD! >>A72C  
 A724 RETURN? >>A798  
 A726 NO, PATHNAME EXPECTED NOW? (BE54)  
 A72A YES, ALL IS WELL >>A762  
 A72C NO, "SYNTAX ERROR" >>A839  
 A72F NON ALPHA FILE NAME, CHECK COMMAND NUMBER (BE53)  
 A732 IS IT "RUN"  
 A734 NO, ERROR >>A72C  
 A736 YES, ITS OK THEN (MIGHT BE "RUN 100") >>A798  
 A739 IN#/PR#, REPARSE COMMAND <AA1B>  
 A73C RETURN FOUND - ERROR >>A72C  
 A73E "A"? (ADDRESS KEYWORD)  
 A740 IF SO, GO PARSE THAT KEYWORD ONLY >>A78C  
 A742 ELSE, ZERO ACCUMULATOR <AB37>  
 A745 CONVERTING ONE BYTE'S WORTH (BCAD)  
 A74A PUT IT IN PR#/IN# SLOT VALUE AREA (BCAE)  
 A74F FOUND SLOT FOR PR#/IN# (BE56)  
 A752 CONVERT SLOT # <A960>  
 A755 ERROR? >>A761  
 A757 GET CONVERTED VALUE (BE6B)  
 A75A >8?  
 A75C NO, ITS OK >>A791  
 A75E YES, "RANGE ERROR"  
 A761 RETURN  
 A762 SECOND PATHNAME EXPECTED?  
 A763 NO >>A787  
 A765 YES, FLUSH TO NON-BLANK <AA3A>  
 A768 NOTHING ELSE ON LINE??? >>A72C  
 A76B DON'T FLUSH ANY BLANKS OUT OF PATHNAME  
 A772 COPY SECOND PATHNAME TO \$281 <AA00>

BASIC Interpreter (BI) -- V1.1 --18 JUN 84 NEXT OBJECT ADDR: A777

ADDR DESCRIPTION/CONTENTS

A777 SAVE IT'S LENGTH (LESS 1) (0280)  
 A77C FOUND PATHNAME1 AND PATHNAME2 (BE56)  
 A780 GET LAST CHARACTER AGAIN <AA3A>  
 A783 IF NOT COMMA OR RETURN, "SYNTAX ERROR" >>A72C  
 A785 RETURN? >>A798  
 A787 NO, COMMA, FLUSH TO NON-BLANK <AA3A>  
 A78A SYNTAX ERROR IF TWO COMMAS IN A ROW >>A72C  
 A78C LOOKUP KEYWORD CHAR AND PARSE ITS VALUE <A8E8>  
 A78F EXIT NOW? >>A761  
 A791 NO, FLUSH TO NON-BLANK <AA3A>  
 A794 SYNTAX ERROR IF COMMA OR RETURN NOT FOUND >>A72C  
 A796 COMMA? YES, GO GET NEXT KEYWORD >>A787  
 A798 GET PARSED SLOT (BE61)  
 A79B MUST BE NON-ZERO >>A75E  
 A79D AND LESS THAN 8  
 A79F OR ELSE - "RANGE ERROR" >>A75E  
 A7A1 CHECK DRIVE TOO (BE62)  
 A7A6 MUST BE EITHER 1 OR 2  
 A7AD IS THIS A DEFERRED COMMAND?  
 A7B0 NO...>>A7BB  
 A7B2 YES, IS A PROGRAM RUNNING? (BE42)  
 A7B5 YES >>A7BB  
 A7B7 NO, "NOT DIRECT COMMAND"  
 A7BA RETURN  
 A7BB EXPECTING NO PATHNAMES? >>A7FD  
 A7BD NO... (BE55)  
 A7C0 ARE S AND D VALID FOR THIS CMD?  
 A7C2 NO >>A7FD  
 A7C4 YES, HAVE WE GOT PATHNAME1? (BE56)  
 A7C8 YES >>A7D3  
 A7CD IS PATHNAME REQUIRED?  
 A7CF YES, "SYNTAX ERROR" >>A839  
 A7D1 NO, OPTIONAL - NO PREFIX FETCH THEN >>A7FD  
 A7D6 DOES PATHNAME1 START WITH A "/"?  
 A7D8 YES, FULLY QUALIFIED >>A7DF  
 A7DA NO, IS THERE A PREFIX ACTIVE? (BF9A)  
 A7DD NO >>A7E8  
 A7DF YES, (BE57)  
 A7E2 SLOT/DRIVE GIVEN WITH THIS COMMAND?  
 A7E4 NO, FORGET IT >>A7FD  
 A7E6 YES, DO WE HAVE PATHNAME ALSO? >>A7F8  
 A7E8 NO,  
 A7EA NULL OUT PATHNAME1 (BCBC)  
 A7F2 MARK THAT WE WILL HAVE ONE SOON (BE56)  
 A7FB ADD PREFIX TO FILENAMES <A83D>  
 A7FB ERROR? >>A83B  
 A7FD GET COMMAND NUMBER (BE53)  
 A800 \*2 AS INDEX INTO TABLE  
 A802 GET ADDRESS OF COMMAND HANDLING ROUTINE (B8E9)  
 A80B AND STORE IT FOR INDIRECT JMP (BCAC)



BASIC Interpreter (BI) -- V1.1 --18 JUN 84 NEXT OBJECT ADDR: A810

ADDR DESCRIPTION/CONTENTS

```

A810 EXTERNAL COMMAND? IF SO GO NOW! >>A836
A812 MY OWN COMMAND, "PREFIX"?
A814 YES, GO NOW >>A836
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 <B7D0>
A826 NO ERRORS I HOPE >>A836
A828 ERROR WAS PATH NOT FOUND?
A82A NO, REAL ERROR - SAY SO >>A83B
A82F CAN WE CREATE PATHNAME1?
A831 YES, OK THEN >>A836
A833 ELSE, "PATH NOT FOUND"
A835 RETURN
A836 GO TO COMMAND HANDLING ROUTINE >>BCAB
    
```

A839 \*\*\*\*\* SYNTAX ERROR \*\*\*\*\*

```

A839 LOAD BI CODE FOR "SYNTAX ERROR"
A83B AND RETURN WITH ERROR CONDITION
A83C RETURN
    
```

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

```

A83D GET SLOT NUMBER (BE61)
A844 PUT SLOT IN HIGH 3 BITS
A846 ADD DRIVE TO TOP BIT AND SHIFT SLOT DOWN (BE62)
A84E ..TO FORM THE UNIT NUMBER (BEC7)
A853 READ THE PATHNAME PREFIX TO $201 (BEC8)
A85D 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 >>A8E6
A878 ELSE, GET LENGTH OF PATHNAME
A87A BUMP IT BY 2 (TO ALLOW FOR /'S)
A882 WITH PREFIX WILL IT EXCEED 64 CHARS?
A887 YES, "SYNTAX ERROR" >>A8E7
A889 NO, UPDATE LENGTH TO INCLUDE PREFIX (BCBC)
A88F ---
A893 AND COPY PATHNAME1 FORWARD TO MAKE ROOM (BCBD)
A89C PUT A "/" AT THE BEGINNING
A8A1 AND AT THE END (BCBD)
A8A4 COPY PREFIX JUST READ TO START OF PATHNAME1 (0200)
A8AA GET COMMAND NUMBER (BE53)
A8AD "OPEN"?
A8AF YES, DONE NOW! >>A8E6
A8B1 "APPEND"?
A8B3 YES, DONE NOW! >>A8E6
    
```

BASIC Interpreter (BI) -- V1.1 --18 JUN 84 NEXT OBJECT ADDR: A8B5

ADDR DESCRIPTION/CONTENTS

```

A8B5 "EXEC"?
A8B7 YES, DONE NOW! >>A8E6
A8B9 ELSE, GET LENGTH OF PATHNAME2 (0280)
A8BE COMBINE THIS WITH PREFIX LENGTH (0201)
A8C1 MORE THAN 64 CHARS?
A8C6 IF SO, "SYNTAX ERROR" >>A8E7
A8C8 UPDATE LENGTH (0280)
A8CB ---
A8CF COPY PATHNAME2 FORWARD TO MAKE ROOM (0281)
A8D8 PUT A "/" IN FIRST
A8DD THEN THE PREFIX AND ANOTHER SLASH (0281)
A8E6 ---
A8E7 DONE!
    
```

A8E8 \*\*\*\*\* KEYWORD LOOKUP \*\*\*\*\*

```

A8E8 ZERO THE ACCUMULATOR <AB37>
A8EB NINE POSSIBLE KEYWORDS IN TABLE
A8ED COMPARE AGAINST EACH (B96B)
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 (BCAE)
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 "V" >>A947
A92C IS THIS KEYWORD PERMITTED? (BE55)
A92F NO, NOT WITH THIS COMMAND ANYWAY >>A923
A931 S OR D?
A933 NO >>A941
A935 YES, ALREADY FOUND IT ON THIS LINE? (BE57)
A938 YES, DON'T CHANGE DRIVE DEFAULT >>A947
A93A ELSE, ASSUME DRIVE = 1
A941 MARK WE HAVE SLOT/DRIVE (BE57)
A947 MARK WE HAVE SLOT/DRIVE (BE57)
    
```

BASIC Interpreter (BI) -- V1.1 --18 JUN 84 NEXT OBJECT ADDR: A954  
 ADDR DESCRIPTION/CONTENTS

A954 AND OFFSET TO VALUE IN STORAGE AREA (BCAE)  
 A957 FLUSH TO NON-BLANK <AA3A>  
 A95A NOTHING ELSE THERE? >>A9B0  
 A95C IS NEXT CHAR A "\$"?  
 A95E YES, GO CONVERT HEX - ELSE, FALL THRU >>A976  
 A960 \*\*\*\*\* CONVERT DECIMAL NUMBER \*\*\*\*\*  
 A960 SAVE LINE INDEX (BE4B)  
 A963 CONVERT/ADD ONE DECIMAL DIGIT TO ACCUM <AA5C>  
 A966 OK.. >>A96C  
 A968 OVERFLOW? THEN "RANGE ERROR" >>A9B3  
 A96A BAD DIGIT? THEN "SYNTAX ERROR" >>A9B0  
 A96C RESTORE LINE INDEX (BE4B)  
 A96F FLUSH TO NEXT NON-BLANK <AA3A>  
 A972 AND GO BACK TO CONVERT NEXT DIGIT >>A960  
 A974 ALL DONE, END OF LINE OR COMMA >>A98F  
 A976 \*\*\*\*\* CONVERT HEX NUMBER \*\*\*\*\*  
 A976 FLUSH TO NEXT NON-BLANK (SKIP "\$") <AA3A>  
 A979 NOTHING LEFT? >>A9B0  
 A97B SAVE LINE INDEX (BE4B)  
 A97E CONVERT HEX DIGIT <AAAE>  
 A981 OK.. >>A987  
 A983 OVERFLOW? THEN "RANGE ERROR" >>A9B3  
 A985 BAD DIGIT? THEN "SYNTAX ERROR" >>A9B0  
 A987 RESTORE LINE INDEX (BE4B)  
 A98A FLUSH TO NEXT NON-BLANK <AA3A>  
 A98D AND GO CONVERT NEXT DIGIT >>A97B  
 A98F \*\*\*\*\* STORE KEYWORD VALUE \*\*\*\*\*  
 A98F HOW MANY BYTES TO CHECK?  
 A994 ALL HAVE BEEN CHECKED? >>A99E  
 A996 NO, INSURE MSB'S OF ACCUM ARE ZERO (BCAF)  
 A999 IF NUMBER IS A SHORT INTEGER >>A9B3  
 A9A1 COPY ACCUM TO PROPER FARM STORAGE CELL (BCAF)  
 A9AB RESTORE LINE INDEX (BE4B)  
 A9AF AND EXIT  
 A9B0 "SYNTAX ERROR" JUMP >>A839  
 A9B3 "RANGE ERROR" JUMP >>A75E  
 A9B6 \*\*\*\*\* STORE KEYWORD VALUE \*\*\*\*\*  
 A9B6 ---  
 A9B8 COPY 3 CHARACTER TYPE TO ACCUM (BCAF)  
 A9BE (COPIED ALL 3?) >>A9C7  
 A9C0 (GET NEXT CHAR IGNORING BLANKS) <AA3A>  
 A9C5 MUST HAVE 3 CHARACTERS! >>A9B0

BASIC Interpreter (BI) -- V1.1 --18 JUN 84 NEXT OBJECT ADDR: A9C7  
 ADDR DESCRIPTION/CONTENTS

A9C7 SAVE LINE INDEX (BE4B)  
 A9CA INITIALIZE NAME INDEX TO ZERO  
 A9CF HAVE ALL 13 BEEN CHECKED?  
 A9D1 YES, NO MATCH >>A9B0  
 A9D4 ELSE, INDEX\*3 (BCAD)  
 A9D8 COMPARE TYPE GIVEN (BCAF)  
 A9DB TO TYPES IN TABLE (B997)  
 A9DE (IGNORE MSB'S)  
 A9DF NO MATCH ALREADY... >>A9E9  
 A9E3 ELSE.  
 A9E5 CHECK ALL THREE CHARS >>A9D8  
 A9E7 THEY ALL MATCH! WE FOUND IT >>A9EE  
 A9E9 NOT THE RIGHT ONE, (BCAD)  
 A9EC GO TRY THE NEXT ONE >>A9CA  
 A9EE REVERSE NAME INDEX  
 A9F5 AND GET TYPE VALUE FROM TABLE (B989)  
 A9F8 STORE IT IN TYPE VALUE STORAGE AREA (BE6A)  
 A9FB RESTORE LINE INDEX (BE4B)  
 A9FF AND EXIT  
 AA00 \*\*\*\*\* COPY PATHNAME2 \*\*\*\*\*  
 AA00 GET NEXT CHARACTER <AA4A>  
 AA03 AND STORE IT INDEXED OFF \$280 (0280)  
 AA07 COMMA?  
 AA09 YES, DONE >>AA37  
 AA0B BLANK?  
 AA0D YES, DONE >>AA37  
 AA0F RETURN?  
 AA11 YES, OUT NOW >>AA48  
 AA13 PATHNAME TOO LONG? (BCAA)  
 AA16 NO, CONTINUE COPYING >>AA00  
 AA18 ELSE, SET NOT-EQUAL CONDITION  
 AA1A AND EXIT  
 AA1B \*\*\*\*\* COPY COMMAND NAME INTO TXTBUF \*\*\*\*\*  
 AA1B SET INDICIES  
 AA1F GET NEXT NON-BLANK <AA4A>  
 AA22 COPY TO TXTBUF (BCBD)  
 AA26 COMMA?  
 AA28 YES, DONE >>AA37  
 AA2A BLANK?  
 AA2C YES, DONE >>AA37  
 AA2E RETURN?  
 AA30 YES, DONE >>AA48  
 AA32 AT MAX LENGTH (8)? (BCAA)  
 AA35 NO, CONTINUE >>AA1F  
 AA37 ELSE, SET NOT-EQUAL CONDITION  
 AA39 AND EXIT

BASIC Interpreter (BI) -- V1.1 --18 JUN 84      NEXT OBJECT ADDR: AA39  
 ADDR      DESCRIPTION/CONTENTS

BASIC Interpreter (BI) -- V1.1 --18 JUN 84      NEXT OBJECT ADDR: AAAA  
 ADDR      DESCRIPTION/CONTENTS

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

AAAA \*\*\*\*\* CONVERT HEX DIGIT AND ADD \*\*\*\*\*

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

AAAAE NUMERIC?  
 AAB0 NO >>AABE  
 AAB4 YES >>AAC4  
 AAB6 NON-NUMERIC, HOW BOUT "A" THRU  
 "F"  
 AABA YES! >>AAC2  
 AABE ---  
 AABF NO, GET OUT NOW  
 AAC1 RETURN  
 AAC2 "A" THRU "F", CONVERT TO \$BA-\$BF  
 AAC4 ISOLATE DIGIT  
 AAC8 SHIFT ACCUM 4 BITS LEFT TO MAKE ROOM <AAD7>  
 AACB (WATCH OUT FOR OVERFLOW) >>AAAA  
 AAD0 OR IN NEW NIBBLE (BCAF)  
 AAD3 AND REPLACE IN ACCUM LSB (BCAF)  
 AAD6 DONE

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

AAAD7 \*\*\*\*\* SHIFT 3 BYTE ACCUM LEFT A BIT \*\*\*\*\*

AAAD7 SHIFT THE THREE BYTE WORK ACCUM (BCAF)  
 AAEB0 RETURN

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>  
 AA84 ---  
 AA85 ACCUM\*4+ACCUM --> ACCUM\*5 (BCAF)  
 AA91 FINALLY, ACCUM\*5\*2 --> ACCUM\*10 <AAD7>  
 AA94 ---  
 AA95 ACCUM OVERFLOW? >>AAAA  
 AA97 NO, ADD NEW DIGIT TO ACCUM (BCAF)  
 AA9A AND STORE IT (BCAF)  
 AA9D NO CARRY? >>AAAD  
 AAA0 GOT CARRY, PROPAGATE IT THRU ACCUM (BCB0)  
 AAAA OVERFLOW ERROR  
 AAAD NORMAL EXIT

AAE1 \*\*\*\*\* SCAN CMD TABLE FOR COMMAND \*\*\*\*\*

AAE1 START WITH LAST COMMAND IN TABLE  
 AAE6 IS IT A "-" COMMAND? (BCBD)  
 AAE8 NOPE >>AAF5  
 AAE9 YES, SPECIAL COMMAND NUMBER (BE53)  
 AAF0 ZERO LENGTH COMMAND STRING (BE52)  
 AAF3 CONTINUE >>AB12  
 AAF5 FIRST COMMANDS IN TABLE ARE 8 CHARS  
 AAF6 GET INDEX TO NEXT NAME (B850)  
 AAFD SAME LENGTH AS LAST NAME? >>AB05  
 AAF0 NO,  
 AB02 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 HAPPILY  
 AB24 RETURN

BASIC Interpreter (BI) -- V1.1 --18 JUN 84      NEXT OBJECT ADDR: AB24  
 -----  
 ADDR    DESCRIPTION/CONTENTS  
 -----

AB25 NOT THE ONE, SKIP TO NEXT (BE52)  
 AB2E IF THERE ARE ANY MORE >>AAFA  
 AB30 ELSE, NO SUCH COMMAND (BE53)  
 AB34 XRETURN THRU \$BE06 VECTOR >>BE06

AB37 \*\*\*\*\* ZERO THREE BYTE ACCUM \*\*\*\*\*  
 AB37 ZERO THE THREE BYTE WORK  
 AB39 ..ACCUMULATOR (BCAF)  
 AB42 RETURN

AB43 \*\*\*\*\* "-" COMMAND \*\*\*\*\*  
 AB43 CHECK FILE TYPE (BE88)  
 AB46 APPLESOFT PROGRAM?  
 AB48 YES, "RUN" IT >>ABB2  
 AB4A BINARY FILE?  
 AB4C YES, "BRUN" IT >>AB8D  
 AB4E TEXT FILE?  
 AB50 NO >>AB55  
 AB52 YES, "EXEC" IT >>B221  
 AB55 SYS, FILE?  
 AB57 YES, GO RUN IT >>AB5D  
 AB59 ELSE, "FILE TYPE MISMATCH"  
 AB5C RETURN

\*\*\*\*\* RUN "SYS" FILE \*\*\*\*\*

AB5D CLOSE ALL OPEN FILES <B4F2>  
 AB60 CLOSE EXEC <B2FB>  
 AB65 LSB OF A\$ IS 00 (BE58)  
 AB68 FREE UP ALL OF BI'S MEMORY (BF6B)  
 AB7B A\$2000 IS WHERE IT WILL LOAD (BE59)  
 AB80 TYPE IS "SYS" (BE6A)  
 AB8A FORCE, T, PATHNAME1, AD PARMS (BE56)  
 AB8D GO DO A STANDARD BRUN >>AE16

AB90 \*\*\*\*\* "CHAIN" COMMAND \*\*\*\*\*  
 AB90 SQUASH VARIABLES UP AGAINST HIMEM <A40D>  
 AB95 SAVE HIMEM (BC7B)  
 AB9C SET NEW HIMEM BELOW COMBINED VARS  
 AB9E LOAD FILE (LEAVE OTHERS OPEN) <AC03>  
 AB4A RESTORE OLD HIMEM  
 AB46 ERROR? >>AC14  
 AB48 NO, CLEAR VARIABLES <D665>  
 ABAB REEXPAND VARIABLES DOWN AGAINST LOMEM <A472>  
 ABB0 THEN GO "RUN" PROGRAM >>ABC7

BASIC Interpreter (BI) -- V1.1 --18 JUN 84      NEXT OBJECT ADDR: ABB0  
 -----  
 ADDR    DESCRIPTION/CONTENTS  
 -----

ABB2 \*\*\*\*\* "RUN" COMMAND \*\*\*\*\*  
 ABB2 NO INPUT FILE ACTIVE NOW  
 ABB7 NO APPLESOFT ERROR NUMBER  
 ABBC GOT PATHNAME1?  
 ABBD NO, ERROR >>ABD5  
 ABBF YES, LOAD PROGRAM <ABFE>  
 ABC2 ERROR? >>AC14  
 ABC4 NO, CLEAR VARIABLES <D665>

ABC7 CLEAR ERROR FLAG  
 ABC9 POSITION TO LINE NUMBER IF GIVEN <AC97>  
 ABCC RESTORE MY INTERCEPTS <9ABD>  
 ABCF CLEAR COMMAND NUMBER ETC., MODE = 4 <ABD5>  
 ABD2 JUMP INTO APPLESOFT TO RUN PROGRAM >>D7D2

ABD5 \*\*\*\*\* CLEAR COMMAND NUMBER ETC. \*\*\*\*\*  
 ABD5 SET NORMAL (NON-INVERSE OR FLASH) <F273>  
 ABDA SEARCH CHARACTER FOR TRACE IS "#" (9F61)  
 ABDF NO COMMAND NUMBER NOW (BE53)  
 ABE2 NO PROMPT  
 ABE6 SET MODE=4 (DEFERRED) <9F76>  
 ABE9 "SYNTAX ERROR" IF THINGS GO WRONG >>A839

ABEC \*\*\*\*\* "LOAD" COMMAND \*\*\*\*\*  
 ABEC LOAD PROGRAM <ABFE>  
 ABEF ERROR? IF NOT, FALL THRU TO WARMSTART >>AC14

ABF1 \*\*\*\*\* WARMDS: WARMSTART BI \*\*\*\*\*  
 ABF1 CLEAR APPLESOFT, RESET POINTERS <D665>  
 ABF4 RESET MODE/SET INTERCEPTS <9A17>  
 ABF9 CURSOR HORIZ. = 0 (START OF LINE)  
 ABFB GO WARMSTART APPLESOFT >>D43F

ABFE \*\*\*\*\* LOAD A PROGRAM \*\*\*\*\*  
 ABFE CLOSE ALL OPEN FILES <B4F2>  
 AC01 ERROR? >>AC14  
 AC03 GO LOAD FILE <AC15>  
 AC06 ERROR? >>AC14  
 AC08 SET LOMEM = ARRAYS = FREESTART  
 AC0A ALL TO END OF PROGRAM LOADED  
 AC14 RETURN

BASIC Interpreter (BI) -- V1.1 --18 JUN 84 NEXT OBJECT ADDR: AC14  
 ADDR DESCRIPTION/CONTENTS

BASIC Interpreter (BI) -- V1.1 --18 JUN 84 NEXT OBJECT ADDR: ACBA  
 ADDR DESCRIPTION/CONTENTS

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 \*\*\*\*\*  
 ---  
 AC62 WAS APPLESOFT PROGRAM SAVED FROM SAME  
 AC64 MEMORY LOCATION? (BEB9)  
 AC73 YES, NOTHING TO DO THEN >>ACBA  
 AC79 ELSE, LOOP THROUGH PROGRAM  
 AC7B ADJUSTING ALL ADDRESSES TO  
 AC7D THE NEW LOAD LOCATION

AC97 \*\*\*\*\* POSITION TO LINE NUMBER \*\*\*\*\*  
 AC97 WAS A LINE NUMBER PARM GIVEN? (BE57)  
 AC9D NO, NEVER MIND >>ACBA  
 AC9F COPY L KEYWORD VALUE TO APPLESOFT'S LINE # (BE68)  
 ACA9 THEN CALL APPLESOFT TO FIND THE LINE <DG1A>  
 ACAF SUBTRACT ONE FROM THE ADDRESS  
 ACB1 AND POINT APPLESOFT'S GETCHR SUBROUTINE  
 ACB3 AT IT (SO NEXT CHAR READ WILL BE FIRST  
 ACB5 CHARACTER ON THE LINE).  
 ACBA RETURN

ACBB \*\*\*\*\* "SAVE" COMMAND \*\*\*\*\*  
 ACBB DOES FILE EXIST ALREADY? >>ACDF  
 ACBD NO, TYPE = BAS  
 ACBF IN T KEYWORD VALUE (BE6A)  
 ACC2 AND MLI LIST (BEB8)  
 ACC7 ALLOW ALL ACCESSES (READ/WRITE/ETC.) (BEB7)  
 ACCC SAVE PROGRAM START ADDRESS IN (BEA5)  
 ACCF AUXID'S (BEB9)  
 ACDA GO CREATE A NEW FILE <AD46>  
 ACDD ERROR? >>AD28

ACDF WRITE ACCESS REQUESTED  
 ACE1 BAS TYPE FILE  
 ACE3 OPEN IT <B194>  
 ACE6 ERROR? >>AD28  
 ACEB SUBTRACT APPLESOFT PTRS TO COMPUTE  
 ACED LENGTH OF PROGRAM.  
 ACEE STORE THIS IN EOF MARK LIST (BEC8)  
 ACFB MSB OF EOF MARK IS 00 (<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) <BE70>  
 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, (BEB9)  
 AD2F EXIT THRU SET FILE INFO ROUTINE >>B7D9

AD32 \*\*\*\*\* "CREATE" COMMAND \*\*\*\*\*  
 AD32 AUXID = 0 (A\$ OR RECLN)  
 AD3D TYPE KEYWORD GIVEN?  
 AD3F YES >>AD46  
 AD43 NO, ASSUME TYPE = DIR (BE6A)  
 AD46 \*\*\* CREATE FILE ENTRY \*\*\* (BE43)  
 AD49 EXEC FILE ACTIVE?  
 AD4C HOW MANY FILES ARE OPEN INCLUDING EXEC? (BE4D)  
 AD4F 8 OR MORE?  
 AD51 YES, ERROR >>AD6E  
 AD56 ELSE, SET TYPE IN MLI LIST (BEA4)  
 AD59 FULL ACCESS (READ/WRITE/ETC.)  
 AD5B KIND = STANDARD FILE  
 AD5D DIR FILE WANTED?

BASIC Interpreter (BI) -- V1.1 --18 JUN 84      NEXT OBJECT ADDR: AD5F  
 -----  
 ADDR    DESCRIPTION/CONTENTS  
 -----

```
AD5F NO >>AD63
AD61 YES, KIND = DIR FILE
AD63 SET ACCESS (BEA3)
AD66 AND KIND (BEA7)
AD6B MLI: CREATE (DON'T COME BACK HERE) >>BE70
AD6E "RAM TOO LARGE" ERROR
AD70 RETURN
```

```
AD71 ***** "RENAME" COMMAND *****
AD71 ---
AD75 SECOND PATHNAME GIVEN?
AD78 IF SO, GO MLI: RENAME >>AD7F
AD7A "SYNTAX ERROR" OTHERWISE >>A839
AD7D ***** "DELETE" COMMAND *****
AD7D SETUP MLI: DELETE CALL TYPE
AD7F EXIT THRU MLI CALL >>BE70
```

```
AD82 ***** "LOCK" COMMAND *****
AD82 GET FILE INFO FOR PATHNAME1 <B7D0>
AD85 GET ACCESS CODES (BEB7)
AD88 TURN OFF ALL...
AD8A BUT READ
AD8F THEN GO SET UPDATED FILE INFO >>B7E7
```

```
AD92 ***** "UNLOCK" COMMAND *****
AD92 GET FILE INFO FOR PATHNAME1 <B7D0>
AD95 TURN ON ALL FILE ACCESSES
AD9D THEN GO SET UPDATED FILE INFO >>B7E7
```

```
ADA0 ***** "PREFIX" COMMAND *****
ADA0 SLOT/DRIVE GIVEN ON COMMAND? (BE57)
ADA6 IF SO, GOT OPERAND ALREADY >>ADAC
ADA8 ELSE, (BE56)
ADAB CHECK FOR PATHNAME1
ADAC AND GO DO MLI: SET PREFIX ...
ADAE IF IT'S THERE >>AD7F
ADB0 ELSE, IS BASIC PROGRAM RUNNING?
ADB2 IF SO, SET PREFIX ACTIVE FLAG >>ADDD1
ADB4 NO, NEW LINE <9FAB>
ADBC END OF NAME YET? >>ADCC9
ADBE NO, COPY NAME IN PATHNAME1 BUFFER (BCBD)
ADC3 TO OUTPUT DEVICE <9FAD>
ADC9 AND SKIP A BLANK LINE <9FAB>
ADD0 DONE
```

BASIC Interpreter (BI) -- V1.1 --18 JUN 84      NEXT OBJECT ADDR: ADD0  
 -----  
 ADDR    DESCRIPTION/CONTENTS  
 -----

```
ADD1 SET PREFIX ACTIVE FLAG
ADD3 SO BASIC CAN READ THE PREFIX (BE46)
ADD7 RETURN
```

```
ADD8 ***** "BSAVE" COMMAND *****
ADD8 PATHNAME1 FOUND? >>AE0E
ADD9 NO, NEW FILE (BE57)
ADD9 AD, L, AND E POSSIBLE
ADDF AD AND EITHER L OR E REQUIRED
ADE1 OR ELSE ERROR >>AE12
ADE6 PUT AD IN CREATE PARAMETER LIST (BEA5)
ADE9 AND IN GET FILE INFO LIST (BEB9)
ADF7 TYPE = BIN ASSUMED (BE6A)
AE00 T KEYWORD GIVEN?
AE02 IF SO, ERROR >>AE12
AE04 GO CREATE THE FILE <AD46>
AE07 ERROR? >>AE14
AE09 GET FILE INFO <B7D0>
AE0C ERROR? >>AE14
AE0E WRITING...
AE10 GO PROCESS LIKE A BLOAD OTHERWISE >>AE25
```

```
AE12 "PATH NOT FOUND" ERROR
AE14 ---
AE15 RETURN
```

```
AE16 ***** "BRUN" COMMAND *****
      (DOES NOT SET MODE=4 SO DOS COMMANDS MAY
      NOT BE ISSUED AS WITH A BASIC PROGRAM)
```

```
AE16 BLOAD IT FIRST <AE23>
AE19 ERROR? >>AE14
AE1B THEN CALL IT <AE20>
AE1E THEN EXIT
AE1F RETURN
AE20 INDIRECT JMP TO BINARY PROGRAM >>BED7
```

```
AE23 ***** "BLOAD" COMMAND *****
AE23 READING...
AE25 TYPE = BIN
AE27 OPEN THE FILE <B194>
AE2A ERROR? >>AE14
AE2C ASSUME USER SPECIFIED AD KEYWORD (BE58)
AE35 IF SO, USE HIS ADDRESS >>AE47
AE37 ELSE, USE AD IN FILE INFO AUXID (BEB9)
AE40 WAS T KEYWORD GIVEN?
AE42 YES, INVALID PARM (ONLY BIN IS LEGAL) >>AE78
AE47 POINT READ/WRITE PARMS TO DATA (BED7)
```

BASIC Interpreter (BI) -- v1.1 --18 JUN 84 NEXT OBJECT ADDR: AE4D  
 ADDR DESCRIPTION/CONTENTS

```

AE4D PICK UP LENGTH FROM L KEYWORD VALUE (BE5F)
AE53 WAS L OR E GIVEN?
AE55 NEITHER >>AE7C
AE57 BOTH?
AE59 YES...NAUGHTY! >>AE78
AE5B E GIVEN?
AE5D NO, MUST BE L >>AE92
AE5F YES... (BE5D)
AE63 COMPUTE L = (E - AD) (BE58)
AE6F PLUS ONE FOR INCLUSIVE RANGE >>AE72
AE72 MAKE SURE NO BORROW OCCURED >>AE92

AE74 OR ELSE, "RANGE ERROR"
AE77 RETURN

AE78 "INVALID PARM" ERROR
AE7B RETURN

AE7E ---
AE7E MLI: GET EOF <BE70>
AE81 ERROR? >>AE90
AE83 GET L (EOF MARK) (BEC8)
AE89 BETTER NOT EXCEED 64K (BECA)
AE8C NO.. >>AE92

AE8E YES, "PROGRAM TOO LARGE"
AE90 ---
AE91 RETURN

AE92 STORE LENGTH TO READ OR WRITE (BED9)
AE9B B KEYWORD GIVEN?
AE9D NO >>AEC4
AEA1 YES, COPY B VALUE TO SET MARK LIST (BE5A)
AEAA ---
AEAC MLI: SET MARK <BE70>
AEB2 NO ERROR? >>AEC4
AEB4 ERROR, RANGE ERROR?
AEB6 NO >>AE90
AEB8 BSAVING (NOT BLOAD/BRUNING)?
AEBA NO >>AE90
AEBE MLI: FORCE EOF FORWARD TO MARK <BE70>
AEC1 AND TRY SET MARK AGAIN >>AEAA
AEC3 RETURN
AEC4 GET COMMAND NUMBER (BE53)
AEC7 ASSUME READ
AEC9 BSAVE?
AECB NO, READ IS CORRECT >>AECF
AECD WRITING
AECF MLI: READ OR WRITE <BE70>
AED2 ERROR? >>AE90
AED4 THEN EXIT THRU CLOSE >>AF94
  
```

BASIC Interpreter (BI) -- v1.1 --18 JUN 84 NEXT OBJECT ADDR: AED4  
 ADDR DESCRIPTION/CONTENTS

```

AED7 ***** "STORE" COMMAND *****
AED7 PATHNAME1 EXISTS? >>AEEB
AED9 NO, T = VAR BY DEFAULT
AEE1 FULL ACCESS (READ/WRITE/ETC.)
AEE6 CREATE THE FILE <AD46>
AEE9 ERROR? >>AF39
AEEB COMPRESS APPLESOFT VARS AGAINST HIMEM <A40D>
AEF4 OPEN "VAR" FILE FOR WRITE <B194>
AEF7 ERROR? >>AF32
AEF9 POINT TO INTERNAL 5 BYTE HEADER BUFFER <AF3A>
AEFC AND WRITE OUT LENGTHS OF VARS <AF9C>
AEFF ERROR? >>AF32
AF01 STORE ADDRESS OF VARS (BC8E)
AF04 IN READ/WRITE PARM LIST (BED7)
AF07 AND FILE INFO AUXID (BEB9)
AF13 GET LENGTH OF VARS (BC91)
AF19 AND WRITE THEM OUT <AF9C>
AF1C ERROR? >>AF32
AF20 MLI: GET MARK <BE70>
AF25 MLI: SET NEW EOF (TRUNCATE IF NECESSARY) <BE70>
AF28 ERROR? >>AF32
AF2A SET FILE INFO WITH AD OF VARS <B7D9>
AF2D ERROR? >>AF32
AF2F CLOSE FILE <AF94>
AF32 ---
AF34 REEXPAND VARS BACK AGAIN <A472>
AF39 RETURN

AF3A ***** SETUP TO READ/WRITE VAR HDR *****
      APPLESOFT VARIABLES HEADER CONSISTS OF:
      2 BYTE LENGTH OF SIMPLE+ARRAY VARIABLES
      2 BYTE LENGTH OF SIMPLE VARIABLES ONLY
      1 BYTE MSB OF HIMEM FOR THESE VARIABLES

AF3A STORE ADDRESS OF 5 BYTE INFO
AF3C IN READ/WRITE PARM LIST (BED7)
AF46 LENGTH = 5
AF48 RETURN

AF49 ***** "RESTORE" COMMAND *****
AF49 TYPE = VAR
AF4B READING
AF4D OPEN THE FILE <B194>
AF50 ERROR? >>AF39
AF52 SET UP TO READ THE HEADER <AF3A>
AF55 READ 5 BYTE HEADER <AF98>
AF58 ERROR? >>AF39
AF5A PICK UP WHERE TO READ IN COMPRESSED VARS (BEB9)
  
```

BASIC Interpreter (BI) -- V1.1 --18 JUN 84 NEXT OBJECT ADDR: AF5D

ADDR DESCRIPTION/CONTENTS

AF5D FROM AUXID (BC8E)
AF63 ADJUST MSB OF THIS BY THE DIFFERENCE
AF66 BETWEEN HIMEM'S (NOW AND WHEN STORED) (BC8D)
AF73 MAKE SURE VARS WON'T OVERLAY PROGRAM
AF75 IF SO, ERROR >>AF90
AF7F COMPUTE LENGTH OF ALL VARS/STRINGS
AF81 (HIMEM-START) (BC8F)
AF85 GO READ COMBINED VARS INTO MEMORY <AF98>
AF88 ERROR? >>AF39
AF8A CLOSE THE FILE <AF94>
AF8D EXIT BY REEXPANDING THE VARS DOWN >>AF32
AF90 "PROGRAM TOO LARGE" ERROR
AF93 RETURN

AF94 \*\*\*\*\* CLOSE FILE \*\*\*\*\*
AF96 SET MLI CLOSE OP CODE
AND GO TO MLI >>AFA4

AF98 \*\*\*\*\* READ/WRITE A RANGE \*\*\*\*\*
AF98 READ MLI OP CODE
AF9A JUMP IN >>AF9E
AF9C WRITE MLI OP CODE
AF9E STORE LENGTH (BEDA)
AFA4 EXIT THRU MLI:READ OR WRITE >>BE70

AFA7 \*\*\*\*\* "PR#" COMMAND \*\*\*\*\*
AFA7 USE CSWL AND OUTVEC
AFAC JUMP TO COMMON CODE >>AFB5

AFAE \*\*\*\*\* "IN#" COMMAND \*\*\*\*\*
AFAE USE KSWL
AFB3 AND INVEC

AFB5 OR IN SLOT GIVEN BY USER (BE6B)
AFB8 \*2 FOR USE AS INDEX INTO TABLE
AFBD WAS SLOT PARAMETER GIVEN?
AFBF NO... >>AFD2
AFC1 YES, (BE57)
AFC4 AD GIVEN? >>AFE7
AFC6 NO, GET INVEC OR OUTVEC FOR THIS SLOT (BE10)
AFC9 AND STORE ON AD KEYWORD VALUE (BE58)
AFD2 VALIDITY CHECK I/O DRIVER <AFF9>
AFD5 NO GOOD? >>AFE6
AFD7 GET INDEX TO CSWL OR KSWL (BCA9)
AFDD AND REPLACE ONE OR THE OTHER WITH (0036)
AFE0 HIS ADDRESS (BE59)

BASIC Interpreter (BI) -- V1.1 --18 JUN 84 NEXT OBJECT ADDR: AFE6

ADDR DESCRIPTION/CONTENTS

AFE6 RETURN
AFE7 VALIDITY CHECK AD KEYWORD VALUE <AFF9>
AFE8 NO GOOD? >>AFF8
AFEC GOOD, COPY VALUE TO INVEC OR OUTVEC (BE59)
AFF8 EXIT BUT DON'T REDIRECT I/O NOW

AFF9 \*\*\*\*\* VALIDITY CHECK I/O DRIVER \*\*\*\*\*
AFF9 \$3A/3B --> NEW HANDLER (FROM AD PARM) (BE58)
B005 IS DRIVER IN MAIN RAM (BELOW \$C000)?
B007 YES >>B01E

B009 NO, RESET I/O CARD ROMS (CFFF)
B00C USE \$3C TO COUNT ITERATIONS
B00E TEST ROM AT USER'S ADDRESS
B014 FOR STABILITY
B018 256 TIMES
B01C MUST BE OK
B01D RETURN
B01E MAIN RAM I/O DRIVER
B020 MUST START WITH A "CLD" INSTRUCTION
B022 OK... >>B01C

B024 ELSE, "NO DEVICE CONNECTED"
B027 RETURN

B028 \*\*\*\*\* "BYE" COMMAND \*\*\*\*\*
B028 CLOSE ANY OPEN FILES <B4F2>
B02B CLOSE EXEC FILE, IF ANY <B2FB>
B030 MLI CALL: <BF00>
B033 QUIT
B034 USE READ PARMLIST BECAUSE QUIT DOESN'T NEED PARMS.

B036 \*\*\*\*\* "CAT" COMMAND \*\*\*\*\*
B036 39 CHARACTERS PER LINE
B038 THEN PROCESS LIKE "CATALOG" >>B03C

B03A \*\*\*\*\* "CATALOG" COMMAND \*\*\*\*\*
B03A 79 CHARACTERS PER LINE
B03C STORE LINE LENGTH (BCB6)
B042 TEST FOR T AND
B044 ...PATHNAME1 GIVEN
B045 GOT T >>B04A
B047 NO T, T=0 (ANY TYPE WILL DO) (BE6A)
B04A GOT PATHNAME1 >>B051
B04C NO PATHNAME1, GET FILE INFO FOR PREFIX <B7D0>
B04F ERROR? >>B0B7
B051 OPEN/READ DIRECTORY HEADER <B14A>



BASIC Interpreter (BI) -- V1.1 --18 JUN 84 NEXT OBJECT ADDR: B054

BASIC Interpreter (BI) -- V1.1 --18 JUN 84 NEXT OBJECT ADDR: B0D7

ADDR DESCRIPTION/CONTENTS

ADDR DESCRIPTION/CONTENTS

B054 ERROR? >>B0B7  
 B056 SKIP TO A NEW LINE <9FAB>  
 B059 FORMAT DIRECTORY'S NAME TO \$201 <B0B8>  
 B05C PRINT \$201 <9F9D>  
 B05F SKIP TO A NEW LINE <9FAB>  
 B062 BLANK \$201 BUFFER <A66C>  
 B067 UNPACK HEADING MESSAGE LINE <9FB0>  
 B06A PRINT IT (40 OR 80 COLUMNS) <9F9D>  
 B06D SKIP TO A NEW LINE <9FAB>  
 B073 ANY FILES IN THIS DIRECTORY? (BCBA)  
 B076 NO >>B0A3  
 B078 YES, READ NEXT ENTRY <B1D1>  
 B07B ERROR? >>B0B7  
 B07D GET TYPE REQUESTED FOR SEARCH (BE6A)  
 B080 ANY TYPE WILL DO? >>B0B7  
 B082 NO, CHECK TYPE AGAINST THIS ENTRY (0269)  
 B085 NOT IT, SKIP IT >>B0B8D  
 B087 ELSE, FORMAT ENTRY TO \$201 <A4C4>  
 B08A AND PRINT \$201 <9F9D>  
 B08D CHECK KEYBOARD (C000)  
 B090 FOR A CONTROL-C  
 B092 IGNORE ANYTHING ELSE >>B09E  
 B094 CONTROL-C, WHAT STATE ARE WE IN? (BE42)  
 B097 DEFERRED >>B0A3  
 B099 NO, IMMEDIATE, RESET KEYBOARD STROBE (C010)  
 B09C AND EXIT RIGHT NOW >>B0A3  
 B09E ELSE, ANY FILES LEFT IN COUNT? (BCBA)  
 B0A1 YES, CONTINUE >>B078  
 B0A3 ELSE, CLOSE DIRECTORY <AF94>  
 B0A6 ERROR? >>B0B7  
 B0A8 SKIP TO A NEW LINE <9FAB>  
 B0AB FORMAT BLOCKS FREE AND IN USE TO \$201 <B0E7>  
 B0AE ERROR? >>B0B7  
 B0B0 PRINT \$201 <9F9D>  
 B0B3 SKIP A LINE <9FAB>  
 B0B7 DONE

B0D7 ...LINE (0200)  
 B0E1 SET \$200 TO MAXIMUM LENGTH  
 B0E6 RETURN  
 B0E7 \*\*\*\*\* FORMAT BLOCKS FREE/INUSE \*\*\*\*\*  
 B0E7 POINT MLI:ONLINE PARMLIST  
 B0E9 TO TXTBUF (PATHNAME1) (BEC8)  
 B0F1 COPY DEVICE NUMBER (UNIT) (BF30)  
 B0F9 MLI: ONLINE <BE70>  
 B0FC ERROR? >>B0B7  
 B101 ISOLATE NAME LENGTH FROM BUFFER  
 B104 BUMP BY ONE TO INCLUDE "/"  
 B105 AND STORE IT AS A PREFIX (BCBC)  
 B10A STORE "/" AS FIRST CHARACTER (BCBD)  
 B10D GET FILE INFO FOR PREFIX <B7D0>  
 B110 ERROR? >>B0B7  
 B112 BLANK \$201 BUFFER <A66C>  
 B117 UNPACK "BLOCKS FREE: BLOCKS USED.." <9FB0>  
 B11A ZERO THE THREE BYTE ACCUM <AB37>  
 B125 CONVERT AUXID (TOTAL BLOCKS) <A62F>  
 B130 CONVERT BLOCKS USED <A62F>  
 B137 BLOCKS FREE = TOTAL BLOCKS (BEBC)  
 B13E ... - BLOCKS USED (BEED)  
 B145 CONVERT BLOCKS FREE <A62F>  
 B149 DONE!  
 B14A \*\*\*\*\* OPEN/READ DIRECTORY HDR \*\*\*\*\*  
 B14A READ ONLY  
 B14E CHECK FILE KIND (BEBB)  
 B151 VOLUME DIRECTORY?  
 B153 NO >>B158  
 B155 YES, TYPE = DIR (BEB8)  
 B158 OPEN THE FILE <B1A0>  
 B15B ERROR? IF NOT, FALL THRU >>B193  
 B15D \*\*\*\*\* READ DIRECTORY HDR \*\*\*\*\*  
 B15D BUFFER IS \$259  
 B169 LENGTH IS \$2B (ONE ENTRY) (BED9)  
 B173 MLI: READ <BE70>  
 B176 ERROR? >>B193  
 B17A COPY ENTRY LENGTH, ENTRIES PER BLOCK, (027C)  
 B17D AND FILE COUNT FROM DIR HDR (BCB7)  
 B183 STORE ENTRY LENGTH IN READ LENGTH NOW (BED9)  
 B188 SET COUNTER TO FIRST ENTRY IN BLOCK (BCBB)  
 B18D MARK = 0 (START OF FILE) (BEC9)  
 B193 RETURN

B0B8 \*\*\*\*\* FORMAT NAME OF DIRECTORY \*\*\*\*\*

B0B8 BLANK \$201 BUFFER <A66C>  
 B0BB FILE NAME IS AT +1 INTO DIR ENTRY  
 B0BD GET NAME LENGTH/TYPE (025D)  
 B0C2 VOLUME DIRECTORY HEADER?  
 B0C4 NO >>B0CA  
 B0C6 YES, START NAME WITH "/" (0200)  
 B0CA ---  
 B0CB ISOLATE NAME LENGTH FROM TYPE  
 B0CD AND SET UP LENGTH TO COPY (0200)  
 B0D2 COPY DIRECTORY NAME TO (0259)

BASIC Interpreter (BI) -- V1.1 --18 JUN 84 NEXT OBJECT ADDR: B193  
 ADDR DESCRIPTION/CONTENTS

BASIC Interpreter (BI) -- V1.1 --18 JUN 84 NEXT OBJECT ADDR: B21D  
 ADDR DESCRIPTION/CONTENTS

B194 \*\*\*\*\* OPEN FILE \*\*\*\*\*  
 A REGISTER = ACCESS BITS  
 X REGISTER = DEFAULT TYPE

B21E \*\*\*\*\* EXTERNAL COMMAND HANDLER \*\*\*\*\*  
 B21E INDIRECT JMP TO XTRNAD VECTOR >>BE50

B194  
 B198 T KEYWORD GIVEN?  
 B19A NO >>B19F  
 B19C YES, USE KEYWORD VALUE INSTEAD (BE6A)  
 B19F ---  
 B1A0 EXISTING FILE OF THIS TYPE? (BEB8)  
 B1A3 NO, ERROR >>B1C9  
 B1A5 CHECK ACCESS REQUESTED (BEB7)  
 B1A8 REQUESTED ACCESS NOT PERMITTED >>B1CD  
 B1AA SET SYSTEM BUFFER IN OPEN PARM LIST (BC88)  
 B1B2 LEVEL = \$0F (BF94)  
 B1B7 MLI: OPEN <BE70>  
 B1BA ERROR? >>B1C8  
 B1BF SAVE REFNUM IN READ/WRITE PARMLIST (BED6)  
 B1C2 AND CLOSE PARMLIST (BEDE)  
 B1C5 AND GET/SET EOF/MARK LIST (BEC7)  
 B1C8 AND EXIT

B221 \*\*\*\*\* "EXEC" COMMAND \*\*\*\*\*  
 B221 IS THIS FILE OPEN ALREADY? <B41F>  
 B224 NO >>B250  
 B226 YES, EXEC CLOSING? (BE4E)  
 B229 NO >>B24C  
 B22B SAVE REFNUM (BEC7)  
 B230 RESET MARK TO ZERO (BEC8)  
 B23B MLI: SET MARK <BE70>  
 B23E ERROR? >>B245  
 B240 GET REFNUM AGAIN (BEC7)  
 B243 GO RESTART THIS EXEC FILE FROM ITS START >>B2C3

\*\*\*\*\* CLOSE EXEC FILE \*\*\*\*\*  
 B245 PRESERVE CALLER'S AREG  
 B246 AND CLOSE THE FILE <B2FB>  
 B24B THEN RETURN WITH ERROR  
 B24C "FILE BUSY" ERROR  
 B24F RETURN

\*\*\*\*\* CONTINUE EXEC SETUP \*\*\*\*\*  
 B250 EXEC ACTIVE? (BE43)  
 B253 NO >>B25A  
 B255 YES, CLOSE IT <B2FB>  
 B258 ERROR? >>B263  
 B25A GET FILE TYPE (BEB8)  
 B25D SHOULD BE TXT  
 B25F IT IS >>B265  
 B261 ELSE, "FILE TYPE MISMATCH"  
 B263 RETURN WITH ERROR  
 B264 RETURN  
 B265 MOVE STRINGS TO MAKE ROOM FOR A BUFFER <A1F5>  
 B268 NO ROOM? >>B263  
 B26C STORE NEW BUFFER ADDRESS IN PARM LIST (BEC8)  
 B275 GET COUNT OF OPEN FILES (BE4D)  
 B278 NO OTHERS CURRENTLY OPEN? >>B29E

B1D1 \*\*\*\*\* READ NEXT DIRECTORY ENTRY \*\*\*\*\*

B1D1 FORCE MARK TO START OF THIS BLOCK (BEC9)  
 B1D9 CHECK ENTRY NUMBER (BCBB)  
 B1DE LAST ENTRY IN THIS BLOCK? (BCB8)  
 B1E1 NO >>B1ED  
 B1E4 YES, ENTRY NOT NEXT TIME (BCBB)  
 B1E7 BUMP MARK TO NEXT BLOCK (BEC9)  
 B1ED ---  
 B1EF MARK POSITIONED TO PROPER ENTRY YET? >>B1F8  
 B1F1 NO, BUMP POINTER TO NEXT ENTRY (BCB7)  
 B1F4 AND CONTINUE IF STILL FIRST PAGE >>B1ED  
 B1F6 JUST ENTERED SECOND PAGE >>B1EA  
 B1F8 ADD 4 TO PTR TO ADJUST FOR BLOCK PREFIX  
 B1FF MLI: SET MARK <BE70>  
 B202 ERROR? >>B21D  
 B206 MLI: READ <BE70>  
 B209 ERROR? >>B21D  
 B20B BUMP ENTRY COUNTER (BCBB)  
 B211 IS THIS ENTRY VALID?  
 B213 NO, SKIP OVER IT >>B1D1  
 B215 DECREMENT FILE COUNT (BCB9)  
 B21D AND RETURN TO CALLER

BASIC Interpreter (BI) --V1.1 --18 JUN 84 NEXT OBJECT ADDR: B278  
 ADDR DESCRIPTION/CONTENTS

BASIC Interpreter (BI) -- V1.1 --18 JUN 84 NEXT OBJECT ADDR: B2FA  
 ADDR DESCRIPTION/CONTENTS

\*\*\*\*\* MAKE EXEC TOPMOST BUFFER \*\*\*\*\*

B27A OTHERS ARE OPEN...  
 B27C OPENCOUNT\*4 (4 PAGES PER BUFFER)  
 B27E ADD THIS TO MY BUFFER TO FIND TOP BUFFER (BC88)  
 B282 SEARCH OPEN FILES TO FIND THE FILE WHICH (BC93)  
 B285 IS USING THIS BUFFER... >>B28B  
 B28A IF IT IS NOT FOUND, BREAK!  
 B28B ---  
 B28C MOVE THAT FILE TO THE NEW BUFFER INSTEAD (BC93)  
 B28F GET THAT FILE'S REFNUM ALSO (BC9B)  
 B297 MLI: SET BUFF <BE70>  
 B29A NO ERRORS? >>B29D  
 B29C IF ERROR, BREAK!  
 B29D ---

\*\*\*\*\* OPEN NEW EXEC FILE \*\*\*\*\*

B29E SET NEW BUFFER ALLOCATION PAGE (BC88)  
 B2A1 SET UP OPEN LIST FOR EXEC TOO (BECF)  
 B2A6 LEVEL = 0 (BF94)  
 B2AB MLI: OPEN (EXEC FILE) <BE70>  
 B2AE NO ERROR? >>B2B7

---  
 B2B0 IF ERROR, FREE BUFFER FIRST <A24C>  
 B2B1 THEN EXIT WITH ERROR  
 B2B6

B2B7 SAVE BUFFNO FOR EXEC (BECF)  
 B2BD AND REFNUM TOO (BED0)

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

B2C3 SAVE READ REFNUM (BED6)  
 B2C6 AND GET/SET REFNUM (BEC7)  
 B2C9 AND NEWLINE REFNUM (BED2)  
 B2CF SET "L" VALUE FROM AUXID (BE5F)  
 B2D8 SAVE PATHNAME/AUXID IN OPEN FILE TABLE <B3EB>  
 B2DD IGNORE MSB FOR END OF LINE CHARS (BED3)  
 B2E2 MLI: SET NEWLINE <BE70>  
 B2E8 WAS "F" OR "R" GIVEN ON COMMAND LINE?  
 B2EA NO >>B2F4  
 B2EC YES, POSITION TO SPECIFIED STARTING PT <B522>  
 B2EF NO ERRORS? >>B2F4  
 B2F1 IF ERROR, GO CLOSE EXEC >>B245  
 B2F4 MARK EXEC ACTIVE  
 B2FA AND RETURN TO CALLER

\*\*\*\*\* CLOSE EXEC FILE \*\*\*\*\*

B2FB EXEC ACTIVE? (BE43)  
 B2FE NO, SKIP IT >>B30B  
 B300 INDICATE EXEC FILE CLOSING (BE4E)  
 B305 PICK UP REFNUM FOR EXEC (BC9B)  
 B308 AND GO CLOSE IT <B4A5>  
 B30B RETURN

\*\*\*\*\* "VERIFY" COMMAND \*\*\*\*\*

B30C FILE NOT FOUND? >>B347  
 B311 FILE FOUND, WAS A PATHNAME1 GIVEN?  
 B313 YES >>B31D  
 B315 NO,  
 B317 PRINT "(C) APPLE COMPUTER..." <9F8C>  
 B31A AND A NEW LINE <9FAB>  
 B31D THEN EXIT  
 B31E RETURN

\*\*\*\*\* FLUSH ALL OPEN FILES \*\*\*\*\*

B31F REFNUM = 0 (ALL FILES)  
 B321 JUMP INTO FLUSH >>B32F

\*\*\*\*\* "FLUSH" COMMAND \*\*\*\*\*

B323 ---  
 B326 WAS PATHNAME1 GIVEN?  
 B328 NO, FLUSH ALL FILES >>B32F  
 B32A ELSE, LOOK UP NAME IN OPEN FILE LISTS <B41F>  
 B32D NOT AN OPEN FILE >>B337  
 B32F SAVE REFNUM IN PARM LIST (BEDE)  
 B334 MLI: FLUSH <BE70>  
 B337 EXIT

\*\*\*\*\* "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 (BI) -- 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  
 B353 WAS "L" KEYWORD GIVEN?  
 B355 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 "TXT"  
 B36B DOES THE FILE ALREADY EXIST? >>B38E  
 B36D NO, "T" GIVEN? IF SO, ERROR >>B347  
 B36F FORCE TYPE = "TXT" (BE88)  
 B374 FULL ACCESS (BE87)  
 B37A COPY "L" KEYWORD VALUE (BE5F)  
 B37D TO CREATE (BEA6)  
 B380 AND SET FILE INFO LISTS (BEBA)  
 B389 GO CREATE THE FILE <AD46>  
 B38C ERROR? >>B349  
 B38E CHECK FILE TYPE (BE88)  
 B391 AGAINST HIS "T" VALUE (BE6A)  
 B394 MISMATCH? >>B343  
 B396 NO, TYPE = TXT?  
 B398 NO >>B3AD  
 B39A YES, GET RECORD LENGTH FROM AUXID (BEBA)  
 B3A3 WAS "L" KEYWORD VALUE GIVEN?  
 B3A5 YES, USE THAT INSTEAD >>B3AD  
 B3A7 OTHERWISE, SAVE AUXID RECORD LEN (BE60)  
 B3AD ALLOCATE A NEW FILE BUFFER <A1F5>  
 B3B0 ERROR? >>B349  
 B3B2 GET BUFFER PAGE NO. (BC88)  
 B3B5 AND STORE IN OPEN LIST (BECF)  
 B3BA LEVEL = 7 (BF94)  
 B3BF MLI: OPEN <BE70>  
 B3C2 NO ERRORS? >>B3CB  
 B3C4 ---  
 B3C5 ERROR, FREE BUFFER FIRST <A24C>  
 B3CA THEN EXIT WITH ERROR CODE  
 B3CB CHECK FILE TYPE AGAIN (BE88)  
 B3CE "DIR" FILE?  
 B3D0 YES >>B3D3  
 B3D2 NO  
 B3D3 ---

BASIC Interpreter (BI) -- V1.1 --18 JUN 84 NEXT OBJECT ADDR: B3D6

ADDR DESCRIPTION/CONTENTS

B3D6 SET DIR FLAG ACCORDINGLY (BE47)  
 B3D9 USING OPEN COUNT AS AN INDEX (BE4D)  
 B3DF STORE BUFFER LOCATION IN OPEN FILE LIST (BC94)  
 B3E5 ALSO, THE REFNUM (BC9C)  
 B3E8 AND BUMP OPEN FILE COUNT AND FALL THRU (BE4D)  
 B3EB \*\*\*\*\* SAVE FILE NAME/RECLN IN TABLE \*\*\*\*\*  
 B3EB MAKE INDEX FROM REFNUM\*32 BYTES  
 B3F1 GET NAME LENGTH (0280)  
 B3F4 OR IN DIR FLAG (BE47)  
 B3F7 AND STORE IN OPEN FILE NAME LIST (BCFE)  
 B3FD NAME > OR = TO 30 BYTES?  
 B3FF NO... >>B403  
 B401 YES, USE 29  
 B403 STORE THAT AS A LOOP COUNTER  
 B408 COPY "L" KEYWORD VALUE TO NAME LIST TOO (BCFF)  
 B411 ---  
 B412 COPY FILE NAME TO NAME LIST (0280)  
 B41B COPY ALL OF NAME, THEN FALL THRU TO EXIT >>B411  
 B41D \*\*\*\*\* "MON" AND "NOMON" COMMANDS \*\*\*\*\*  
 B41D IGNORE THESE COMMANDS AND  
 B41E RETURN TO CALLER  
 B41F \*\*\*\*\* LOOKUP OPEN FILENAME \*\*\*\*\*  
 (RETURNS REFNUM OF OPEN FILE)  
 B41F ---  
 B422 WAS PATHNAME1 GIVEN?  
 B424 YES >>B42A  
 B426 NO, "SYNTAX ERROR"  
 B429 EXIT WITH ERROR  
 B42A ANY FILES CURRENTLY OPEN? (BE4D)  
 B42D NO, CAN'T FIND IT THEN >>B448  
 B42F YES, CLEAR EXEC FILE CLOSING FLAG (BE4E)  
 B432 STORE FILE COUNT AS LOOP COUNTER  
 B434 GET NEXT REFNUM (BC9B)  
 B437 COMPARE FILENAMES <B462>  
 B43A NOT THE ONE? >>B443  
 B43C ELSE, WE'VE GOT IT!  
 B43E PICK UP APPROPRIATE REFNUM (BC9B)  
 B441 ---  
 B442 AND RETURN WITH IT  
 B443 ELSE, NOT IT, TRY NEXT ONE  
 B446 AND CONTINUE LOOPING >>B432

BASIC Interpreter (BI) -- V1.1 --18 JUN 84 NEXT OBJECT ADDR: B446  
 ADDR DESCRIPTION/CONTENTS

B448 CAN'T FIND IT, IS EXEC ACTIVE? (BE43)  
 B44B NO, THEN WE MUST GIVE UP >>B45E  
 B450 IS HE LOOKING FOR EXEC FILE? <B462>  
 B453 NO, GIVE UP >>B45E  
 B457 YES, EXEC FILE CLOSING (BE4E)  
 B45C AND RETURN WITH EXEC'S REFNUM >>B43E  
 B45E "FILE NOT OPEN" ERROR  
 B461 RETURN WITH ERROR CODE

B462 \*\*\*\*\* COMPARE FILENAMES \*\*\*\*\*

B462 REFNUM\*32 FOR FILENAME INDEX  
 B468 PICK UP DIR FLAG FROM THIS ENTRY (BCFE)  
 B470 SAME LENGTH AS HIS FILENAME? (0280)  
 B473 NO, CAN'T BE IT THEN >>B49B  
 B476 MAKE SURE LENGTH DOES NOT EXCEED 29  
 B47A IF IT DOES, ONLY LOOK AT FIRST 29  
 B47C USE \$3A AS LOOP COUNTER  
 B481 COPY "L" OF THIS FILE TO KEYWORD (BCA4)  
 B48A ---  
 B48B COMPARE NAMES (0280)  
 B491 NO MATCH? EXIT WITH Z FLAG CLEAR >>B498  
 B498 MATCH, EXIT WITH Z FLAG SET

B499 \*\*\*\*\* "CLOSE" COMMAND \*\*\*\*\*

---  
 B499 PATHNAME1 GIVEN?  
 B49E NO, CLOSE ALL FILES >>B4F2  
 B4A0 YES, LOOK IT UP IN OPEN FILE TABLES <B41F>  
 B4A3 NOT FOUND? >>B441  
 B4A5 FOUND IT, STORE REFNUM IN CLOSE LIST (BEDE)  
 B4AB MARK BUFFER PAGE FREE (BC88)  
 B4AE EXEC CLOSING? (BE4E)  
 B4B1 YES...NO NEED TO COMPRESS LISTS >>B4CF  
 B4B3 GET OPEN COUNT (LAST OPENED FILE NO.) (BE4D)  
 B4B7 SWAP BUFFERS (BC93)  
 B4C5 AND REFNUMS WITH THE LAST OPENED FILE (BC9B)  
 B4CF ---  
 B4D1 LEVEL = 0 (BF94)  
 B4D6 MLI: CLOSE <BE70>  
 B4D9 ERROR? >>B502  
 B4DB RELEASE THE BUFFER <A24C>  
 B4DE EXEC FILE CLOSING? (BE4E)  
 B4E1 NO >>B4EE  
 B4E6 YES, EXEC NO LONGER ACTIVE (BE43)  
 B4E9 AND NO LONGER CLOSING (BE4E)  
 B4ED RETURN TO CALLER

BASIC Interpreter (BI) -- V1.1 --18 JUN 84 NEXT OBJECT ADDR: B4ED  
 ADDR DESCRIPTION/CONTENTS

B4EE DROP OPEN FILE COUNT (BE4D)  
 B4F1 AND EXIT

B4F2 \*\*\*\*\* CLOSE ALL OPEN FILES \*\*\*\*\*

B4F2 ANY FILES OPEN? (BE4D)  
 B4F5 NO >>B503  
 B4F7 YES, EXEC NOT CLOSING (BE4E)  
 B4FD CLOSE LAST FILE OPENED <B4A5>  
 B500 IF THAT WORKS, START ALL OVER AGAIN >>B4F2  
 B502 EXIT WHEN ALL ARE CLOSED

---  
 B503 SET CLOSE REFNUM TO ZERO (ALL FILES) (BEDE)  
 B50A LEVEL = 7 (LEVEL 0 FILES ALREADY CLOSED) (BF94)  
 B50F EXIT THRU MLI: CLOSE >>BE70

B512 \*\*\*\*\* "POSITION" COMMAND \*\*\*\*\*

B512 LOOKUP NAME OF FILE <B41F>  
 B515 NOT OPEN? >>B57F  
 B517 SET REFNUM IN READ/WRITE PARMLIST (BED6)  
 B51A AND SET NEWLINE LIST (BED2)  
 B51D DIR FILE? (BE47)  
 B520 YES, GET OUT RIGHT NOW! >>B580

"F" OR "R" GIVEN? (BE57)  
 B527 NO, INVALID PARM >>B57D  
 B529 BOTH GIVEN?  
 B52B YES, INVALID PARM >>B57D  
 B52D JUST "R" GIVEN?  
 B52F NO, JUST "F" >>B53D  
 B531 JUST "R", COPY "R" VALUE TO "F" (BE65)  
 B534 ("R" AND "F" ARE ALIASES) (BEG3)  
 B53D SET COUNT TO 239. (MAXIMUM LINE LEN)  
 B54C BUFFER IS AT \$200 (BED8)  
 B54F ---  
 B551 NEW LINE CHAR IS EITHER \$0D OR \$08 (BED3)  
 B556 MLI: SET NEWLINE <BE70>  
 B559 ERROR? >>B57F

\*\*\*\*\* SKIP LINES BY READING THEM \*\*\*\*\*

---  
 B55B "F" = 0? (BE64)  
 B55E YES, DONE >>B569  
 B564 ELSE...  
 B566 MLI: READ NEXT FIELD (LINE) <BE70>  
 B569 ERROR? >>B57F  
 B56E DECREMENT "F" VALUE BY ONE

BASIC Interpreter (BI) -- V1.1 --18 JUN 84      NEXT OBJECT ADDR: B57B  
 -----  
 ADDR    DESCRIPTION/CONTENTS  
 -----

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 (BE66)  
 B5A6 IF LOW BIT OFF, NO ADD >>B5BF  
 B5A9 ADD ONE INSTANCE OF RECLEN TO MARK (BCAF)  
 B5B8 OVERFLOW? >>B5D2  
 B5BD ACCUM OVERFLOW? >>B5D2  
 B5BF SCALE ACCUM (MULTIPLIER) UP BY 2 (BCAF)  
 B5C8 IF "R" NON ZERO... (BE65)  
 B5CE CONTINUE LOOPING >>B59E  
 B5D1 ELSE, EXIT TO CALLER

B5D2 "RANGE ERROR"  
 B5D5 RETURN  
 B5D6 \*\*\*\*\* "READ" COMMAND \*\*\*\*\*  
 B5D6 LOOK UP FILE NAME <B41F>  
 B5D9 NOT OPEN? >>B62B  
 B5DB ITS OPEN, STORE REFNUM IN READ/WRITE... (BED6)  
 B5DE GET/SET... (BEC7)  
 B5E1 AND SET NEWLINE PARMLISTS (BED2)  
 B5E4 DIR FILE? (BE47)  
 B5E7 YES, SPECIAL HANDLING REQUIRED >>B62C  
 B5E9 NO, PRE-POSITION FOR "B", "F", OR "R" <B666>  
 B5EC ERROR POSITIONING? >>B62B  
 B5EE ASSUME "L" = 239.  
 B5F5 "L" GIVEN?  
 B5F7 NO >>B60C  
 B5F9 YES, USE HIS "L" VALUE (BE5F)  
 B5FF UNLESS ITS >256 >>B661  
 B603 OR >239. >>B661  
 B607 DOUBLE QUOTE IT SO COMMAS COME THRU (0200)  
 B60A READ INTO \$201  
 B60C IF NO "L", READ TO \$200 (BED7)  
 B612 NL CHAR = \$0D/\$0D (OR NONE IF "L") (BED3)  
 B621 MLI: SET NEWLINE <BE70>  
 B624 ERROR? >>B62B  
 B626 ---

BASIC Interpreter (BI) -- V1.1 --18 JUN 84      NEXT OBJECT ADDR: B628  
 -----  
 ADDR    DESCRIPTION/CONTENTS  
 -----

B628 MARK INPUT "READ" FILE ACTIVE (BE44)  
 B62B AND RETURN  
       \*\*\*\*\* READ DIR FILE \*\*\*\*\*  
 B62C SET READ/WRITE LIST REFNUM (BED6)  
 B634 AND GET/SET LIST REFNUM (BEC7)  
 B63F READING TO \$259 (BED7)  
 B63E INIT CAP FLAG TO FIRST LINE VALUE (BE4F)  
 B644 "R" GIVEN?  
 B647 NO, DONE >>B626  
 B64B YES, ZERO OUT MARK (BEC8)  
 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 "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 >>B687  
 B682 SKIP LINES UNTIL "F" = 0 <B53D>  
 B685 ERROR? >>B6B0  
 B687 "B" GIVEN? (BE57)  
 B68C NO >>B6AF  
 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 ---  
 B6AA MLI: SET MARK <BE70>  
 B6AD ERROR? >>B6B0  
 B6AF ---  
 B6B0 ---  
 B6B2 EXIT TO CALLER

BASIC Interpreter (BI) -- V1.1 --18 JUN 84      NEXT OBJECT ADDR: B6B2  
 ADDR    DESCRIPTION/CONTENTS

B6B3 \*\*\*\*\* "WRITE" COMMAND \*\*\*\*\*  
 B6B3 LOOKUP OPEN FILE NAME <B41F>  
 B6B6 NOT AN OPEN FILE? >>B6C8  
 B6B8 STORE READ/WRITE REFNUM (BED6)  
 B6BB AND GET/SET REFNUM (BEC7)  
 B6BE AND NEWLINE REFNUM IN PARM LISTS (BED2)  
 B6C1 DIR FILE? (BE47)  
 B6C4 NO, OK >>B6CA  
 B6C6 YES, "FILE LOCKED" ERROR  
 B6C8 ---  
 B6C9 EXIT TO CALLER  
 B6CA DATA BUFFER AT \$200  
 B6D4 PRE-POSITION FOR "B", "F", AND "R" <B666>  
 B6D7 NO ERRORS? >>B6ED  
 B6D9 WAS ERROR A RANGE ERROR?  
 B6DB NO, REAL ERROR >>B6C8  
 B6DD YES, MY RANGE ERROR OR MLI'S?  
 B6DF MINE... >>B6C8  
 B6E1 MLI'S...SET EOF FARTHER INTO FILE  
 B6E3 MLI: SET EOF <BE70>  
 B6E6 ERROR? >>B6C8  
 B6E8 AND THEN TRY AGAIN TO SET MARK <B676>  
 B6EB ERROR? THEN I GIVE UP >>B6C8  
 B6ED BUFFER IS AT HIMEM  
 B6F9 INDICATE OUTPUT "WRITE" FILE ACTIVE (BE45)  
 B6FD RETURN TO CALLER  
 B6FE \*\*\*\*\* "APPEND" COMMAND \*\*\*\*\*

B6FE \*\*\*\*\* "APPEND" COMMAND \*\*\*\*\*  
 B6FE ---  
 B6FF LOOK UP NAME IN OPEN FILE LIST <B41F>  
 B702 FOUND IT? >>B710  
 B705 NO, OPEN IT FIRST <B338>  
 B708 ERROR? >>B71E  
 B70A NO, REFNUM NON-ZERO? (BED0)  
 B70D YES, OK >>B711  
 B70F ELSE, BREAK!!!  
 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: B71A  
 ADDR    DESCRIPTION/CONTENTS

B71C YES, "FILE LOCKED"  
 B71E ---  
 B71F EXIT TO CALLER  
 B720 PICK UP "L" VALUE (BE5F)  
 B729 DID USER SPECIFY ONE?  
 B72B YES...>>B733  
 B72D NO, USE FILE'S CURRENT "L" VALUE (BEB9)  
 B733 ---  
 B738 COMPUTE REENUM\*32 FOR INDEX INTO  
 B739 FILE NAME TABLE  
 B73E SAVE CURRENT "L" VALUE IN OPEN FILE (BCFF)  
 B741 NAME TABLE AND IN CURRENT RELEN (BCA4)  
 B74D MLI: GET EOF <BE70>  
 B750 ERROR? >>B71E  
 B752 IS "L" VALUE < 2? (NO SPECIFIC "L") (BCA5)  
 B755 NO >>B75E  
 B75C YES >>B763  
 B75E NO, FORCE TO RECORD BOUNDARY <B766>  
 B761 ERROR? >>B71E  
 B763 ELSE, GO SET EOF=MARK/OUTPUT FILE ACTIVE >>B6E1  
 B766 \*\*\*\*\* FORCE TO EVEN RECORD BOUNDARY \*\*\*\*\*  
 (FIND RECORD NUMBER OF THIS POSITION)

B766 \*\*\*\*\* FORCE TO EVEN RECORD BOUNDARY \*\*\*\*\*  
 (FIND RECORD NUMBER OF THIS POSITION)  
 ---  
 B766 COPY EOF TO ACCUM (BEC7)  
 B771 CLEAR MSB'S (BCB2)  
 B777 GET READY FOR A 24 BIT DIVIDE  
 B779 DIVIDE EOF BY... <AAD7>  
 B786 RECORD LENGTH (BCA4)  
 B79B ---  
 B7A1 WAS THERE A REMAINDER? (BCB3)  
 B7A5 NO, OK...>>B7CF  
 B7AB YES, CURRENT RECORD LEN LESS REMAINDER (BCB2)  
 B7B8 PLUS OLD EOF MARK (BEC8)  
 B7C2 GIVES NEW EOF ON AN EVEN RECORD BOUNDARY (BEC9)  
 B7CD "RANGE ERROR" POSSIBLE IF OVERFLOW OCCURS  
 B7CF RETURN TO CALLER  
 B7D0 \*\*\*\*\* GET FILE INFO \*\*\*\*\*  
 B7D0 SET NUMBER OF PARMS (10)  
 B7D5 MLI CODE FOR GET FILE INFO  
 B7D7 GO DO IT >>B7EE

BASIC Interpreter (BI) -- V1.1 --18 JUN 84 NEXT OBJECT ADDR: B7D7
ADDR DESCRIPTION/CONTENTS

BASIC Interpreter (BI) -- V1.1 --18 JUN 84 NEXT OBJECT ADDR: B859
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 >>BE70
B7F1 \*\*\*\*\* BI I/O INDIRECTION VECTORS \*\*\*\*\*
B7F1 DOSOUT VECTOR >>BE38
B7F4 DOSIN VECTOR >>BE3A
B7F7 \*\*\*\*\* STATE I/O VECTORS TABLE \*\*\*\*\*
B7F7 IMMEDIATE MODE (STATE=0) CSWL/KSWL
B7FB DEFERRED MODE (STATE=4) CSWL/KSWL
B7FF (STATE=8) CSWL/KSWL
B803 (STATE=C) CSWL

NAME OF THE GIVEN LENGTH (NEXT WILL BE ONE BYTE LONGER).
B859 01 IN# 02 PR# 03 CAT
B85C 04 FRE 05 BYE 06 RUN
B85F 07 BRUN 08 EXEC 09 LOAD
B862 0A LOCK 0B OPEN 0C READ
B865 0D SAVE 0E BLOAD 0F BSAVE
B868 10 CHAIN 11 CLOSE 12 FLUSH
B86B 13 NOMON 14 STORE 15 WRITE
B86E 16 APPEND 17 CREATE 18 DELETE
B871 19 PREFIX 1A RENAME 1B UNLOCK
B874 1C VERIFY 1D CATALOG 1E RESTORE
B877 1F POSITION
B878 'BSAVERIFYBLOADDELETEBECATALOGE'
B898 'NWRITECREATEFRESTORENAMESBRUNLO'
B8B8 'CKCHAIN#FLUSHREADPOSITIONMONPR#'
B8D8 'PREFIXCLOSEAPPEND'

B805 \*\*\*\*\* SYSTBL \*\*\*\*\*
LSB'S OF MLI CALL PARAMETER LISTS IN THE BI GLOBAL PAGE (\$BEXX)

B8E9 \*\*\*\*\* COMMAND HANDLER ADDRESS TABLE \*\*\*\*\*
ADDRESSES OF THE COMMAND HANDLER ROUTINES FOR EACH COMMAND IN THE ORDER GIVEN ABOVE.

B805 CREATE: \$A0 DESTROY: \$AC RENAME: \$AF
B808 SFI: \$B4 GFI: \$B4 ONLINE: \$C6
B80B SFX: \$AC GPFX: \$AC OPEN: \$CB
B80E NEWLINE:\$D1 READ: \$D5 WRITE: \$D5
B811 CLOSE: \$DD FLUSH: \$DD SMARK: \$C6
B814 GMARK: \$C6 SEOF: \$C6 GEOF: \$C6
B817 SBUF: \$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

(EXTERNAL)
B8E9 IN#
B8EB PR#
B8ED CAT
B8EF FRE
B8F1 BYE
B8F3 RUN
B8F5 BRUN
B8F7 EXEC
B8F9 LOAD
B8FB LOCK
B8FD OPEN
B8FF READ
B901 SAVE
B903 BLOAD
B905 BSAVE
B907 CHAIN
B909 CLOSE
B90B FLUSH
B90D NOMON
B90F STORE
B911 WRITE
B913 APPEND
B915 CREATE
B917 DELETE
B919 PREFIX
B91B RENAME
B91D

B819 FIRST IS \$80 (END)
B823 CALL
B833 TRACE, NOTRACE, NORMAL
B837 INVERSE, FLASH
B83F RESUME
B843 LET, IF
B853 PRINT, LIST

B859 \*\*\*\*\* COMMAND NAME TABLES \*\*\*\*\*
OFFSETS TO LAST CHARACTER OF EACH COMMAND NAME IN THE COMMAND NAME TABLE BELOW.
COMMANDS ARE ARRANGED ACCORDING TO LENGTH WITH THREE BYTE NAMES FIRST. IF THE MSB OF AN INDEX IS ON, THEN THIS IS THE LAST



BASIC Interpreter (BI) -- V1.1 --18 JUN 84 NEXT OBJECT ADDR: B91F

BASIC Interpreter (BI) -- V1.1 --18 JUN 84 NEXT OBJECT ADDR: B949

ADDR DESCRIPTION/CONTENTS

ADDR DESCRIPTION/CONTENTS

B91F UNLOCK
B921 VERIFY
B923 CATALOG
B925 \*RESTORE
B927 POSITION
B929 "-" COMMAND

B949 CHAIN
B94B CLOSE
B94D FLUSH
B94F NOMON
B951 STORE
B953 WRITE
B955 APPEND
B957 CREATE
B959 DELETE
B95B PREFIX
B95D RENAME
B95F UNLOCK
B961 VERIFY
B963 CATALOG
B965 RESTORE
B967 POSITION
B969 "-"

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.
8000 = FETCH PREFIX, PATHNAME OPTIONAL
4000 = SLOT (FOR PR# OR IN#)
2000 = DEFERRED COMMAND ONLY
1000 = FILENAME IS OPTIONAL
0800 = IF FILE NOT FOUND, CREATE IT
0400 = "t" (FILE TYPE) PERMITTED
0200 = PATHNAME2 (RENAME) PERMITTED
0100 = PATHNAME1 EXPECTED
0080 = "A" (ADDRESS) PERMITTED
0040 = "B" (BYTE) PERMITTED
0020 = "E" (END ADDRESS) PERMITTED
0010 = "L" (LENGTH) PERMITTED
0008 = "Q" (LINE NO.) PERMITTED
0004 = "S" AND/OR "D" (SLOT/DRIVE)
0002 = "F" (FIELD) PERMITTED
0001 = "R" (RECORD) PERMITTED
("V" IS IGNORED)

B96B \*\*\*\*\* KEYWORD NAME TABLE \*\*\*\*\*

B96B 'ABELSDFRV@'

B975 \*\*\*\*\* KEYWORD BIT POSITION TABLE \*\*\*\*\*
BIT POSITIONS IN PERMITTED PARAMS TABLE
FOR EACH KEYWORD IN THE ORDER GIVEN IN
NAME TABLE. "V" IS 00 (NOT USED)

B975 ---

C P S D F N E I A A | | | | | | | | | | | | | |
O F L E N E | A A | | | | | | | | | | | | | |
M X O F O W | T H | | | | | | | | | | | | | |
M | T E P F | H H | | | | | | | | | | | | | |
N | | R T I | 2 1 | | | | | | | | | | | | | |
D | | | | | | | | | | | | | | | | | | | | |

B97F \*\*\*\*\* KEYWORD SIZE/OFFSET TABLE \*\*\*\*\*
LOW 2 BITS - SIZE-1 OF VALUE IN BYTES
HIGH 6 BITS- OFFSET TO LAST BYTE OF VALUE
FROM \$BE58

B97F A: 2 BYTES AT +1
B980 B: 3 BYTES AT +4
B981 E: 2 BYTES AT +6
B982 L: 2 BYTES AT +8
B983 S: 1 BYTE AT +9
B984 D: 1 BYTE AT +A
B985 F: 2 BYTES AT +C
B986 R: 2 BYTES AT +E
B987 V: 1 BYTE AT +10 (IGNORED)
B988 @: 2 BYTES AT +11

B92B IN#
B92D PR#
B92F CAT
B931 FRE
B933 BYE
B935 RUN
B937 BRUN
B939 EXEC
B93B LOAD
B93D LOCK
B93F OPEN
B941 READ
B943 SAVE
B945 BLOAD
B947 BSAVE

B989 \*\*\*\*\* FILE TYPES TABLES \*\*\*\*\*
FILE TYPE CODES, GIVEN IN INVERSE ORDER
TO FILE TYPE NAMES WHICH FOLLOW.

BASIC Interpreter (BI) -- V1.1 --18 JUN 84      NEXT OBJECT ADDR: B989  
 -----  
 ADDR    DESCRIPTION/CONTENTS

B989 \$FF = "SYS"  
 B98A \$FE = "REL"  
 B98B \$FD = "VAR"  
 B98C \$FC = "BAS"  
 B98D \$FB = "IVR"  
 B98E \$FA = "INT"  
 B98F \$F0 = "CMD"  
 B990 \$0F = "DIR"  
 B991 \$06 = "BIN"  
 B992 \$04 = "TXT"  
 B993 \$EF = "PAS"  
 B994 \$1A = "AWP"  
 B995 \$1B = "ASP"  
 B996 \$19 = "ADB"

B997 ---  
 B997 'ADBASPAPPASTXBINDIRCMDINTIVRBASVARRELSYS'

B9C1 \*\*\*\*\* MONTH TABLE \*\*\*\*\*

B9C1 'JANFEBMARAPR MAYJUNJUL AUGSEP OCTNOVDEC'  
 B9E5 '<NO DATE>'

B9EE \*\*\*\*\* MLIERTBL \*\*\*\*\*  
 MLI ERROR CODES WHICH HAVE BI EQUIVALENTS

B9EE ---

BA01 \*\*\*\*\* BIERTBL \*\*\*\*\*  
 BI EQUIVALENTS TO MLI ERROR CODES ABOVE  
 (IF MLI CODE NOT FOUND, MAPS TO LAST CODE  
 IN THIS TABLE, \$08 "I/O ERROR")

BA01 ---

BA15 \*\*\*\*\* INDEXES TO PACKED MESSAGES \*\*\*\*\*  
 BY BI ERROR NUMBER

BA15 ---

BA29 \*\*\*\*\* COMMON LETTERS IN MESSAGES \*\*\*\*\*

BA29 ---  
 BA29 'ACDEFILMORTU '

BASIC Interpreter (BI) -- V1.1 --18 JUN 84      NEXT OBJECT ADDR: BA38  
 -----  
 ADDR    DESCRIPTION/CONTENTS

BA38 \*\*\*\*\* LESS COMMON LETTERS \*\*\*\*\*

BA38 ---  
 BA39 'BGHKPSVWXY/().:.'

BA48 \*\*\*\*\* PACKED MESSAGES \*\*\*\*\*

BA48 "COPYRIGHT APPLE COMPUTER"

BA58 " NAME "  
 BA5B TAB(\$10)  
 BA5D "TYPE BLOCKS "  
 BA66 TAB(\$1E)  
 BA68 "MODIFIED"  
 BA6C TAB(\$2F)  
 BA6E "CREATED "  
 BA72 TAB(\$40)  
 BA74 "ENDFILE SUBTYPE"

BA7E "BLOCKS FREE:"  
 BA86 TAB(\$16)  
 BA88 "BLOCKS USED:"  
 BA91 TAB(\$2C)  
 BA93 "TOTAL BLOCKS:"

BA9C "RANGE ERROR"      ERROR=\$2

BAA3 "NO DEVICE CONNECTED"      ERROR=\$3

BAAE "WRITE PROTECTED"      ERROR=\$4

BAB7 "END OF DATA"      ERROR=\$5

BABD "PATH NOT FOUND"      ERROR=\$6,\$7

BACU

BAC6 "I/O ERROR"      ERROR=\$8

BACC "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      NEXT OBJECT ADDR: BAF8  
 -----  
 ADDR    DESCRIPTION/CONTENTS  
 -----

BAFC "PROGRAM TOO LARGE"                    ERROR=\$E  
 BB07 "NOT DIRECT COMMAND"                ERROR=\$F  
 BB11 "SYNTAX ERROR"                      ERROR=\$10  
 BB19 "DIRECTORY FULL"                     ERROR=\$11  
 BB21 "FILE NOT OPEN"                      ERROR=\$12  
 BB29 "DUPLICATE FILE NAME"               ERROR=\$13  
 BB34 "FILE BUSY"                          ERROR=\$14  
 BB3B "FILE(S) STILL OPEN"                ERROR=\$15  
 BB40

BB47 \*\*\*\*\* VARIABLES \*\*\*\*\*  
 BB47 NUMBER OF PAGES TO ALLOCATE/FREE  
 BB48 NOT USED  
 BB49 TOP OF BUFFERS FOR GARBAGE COLLECTION  
 BB4A BOTTOM OF BUFFERS

BB4B \*\*\*\*\* \$BB4B-\$BC7A NOT USED \*\*\*\*\*  
 BB4B NOT USED  
 BC7B \*\*\*\*\* VARIABLES \*\*\*\*\*

BC7B SAVED HIMEM VALUE DURING CHAIN LOAD  
 \*\*\*\*\* GARBAGE COLLECT MARKED GC: \*\*\*\*\*  
 BC7C GC: HIRANGE - WORKSAREASIZE  
 BC7D GC: WORKAREA MSB  
 BC7E GC: NUMBER OF PAGES IN WORKAREA  
 BC7F GC: LORANGE (START OF STRINGS TO COPY)  
 BC80 GC: HIRANGE (END OF STRINGS TO COPY)  
 BC81 ARRAYS START LSB  
 BC82 ARRAYS ENDING MSB+1  
 BC83 GC: START OF STRING AREA (ALSO PGM START)  
 BC85 GC: END OF STRING AREA  
 BC87 MSB ADJUST FACTOR FOR STRING POINTERS  
 BC88 PAGE FOLLOWING BLOCK BUFFER  
 \*\*\*\*\* STORED VARIABLES FILE HEADER \*\*\*  
 BC89 COMBINED LEN OF SIMPLE/ARRAY VARS  
 BC8B LEN OF SIMPLE VARS ONLY  
 BC8D HIMEM WHEN VARS WERE COMBINED  
 \*\*\*\*\*  
 BC8E POINTER TO COMBINED VARIABLES/STRINGS  
 BC90 LENGTH OF COMBINED VARIABLES/STRINGS

BASIC Interpreter (BI) -- V1.1 --18 JUN 84      NEXT OBJECT ADDR: BC92  
 -----  
 ADDR    DESCRIPTION/CONTENTS  
 -----

BC92 LENGTH OF STRINGS ONLY  
 BC94 OPEN FILES' BUFFER MSBS  
 BC9B OPEN EXEC FILE BUFFER MSB  
 BC9C OPEN FILES' REFERENCE NUMBERS  
 BCA3 OPEN EXEC FILE REFNUM  
 BCA4 CURRENT RECORD LENGTH  
 BCA6 NOT USED  
 BCA9 CHARACTER TO FLUSH WHEN PARSING (BLANK)  
 BCAB MAXIMUM LENGTH TO PARSE  
 BCAD ADDRESS OF COMMAND HANDLING ROUTINE  
 BCAD SIZE OF KEYWORD VALUE -1 IN BYTES  
 BCAD OFFSET INTO KEYWORD PARMS TO LAST BYTE  
 BCAD GENERAL PURPOSE 4 BYTE ACCUMULATOR  
 BCB3 MONTH  
 BCB4 DAY  
 BCB5 YEAR  
 BCB6 ERROR MSG LEN OR LINE LEN FOR CAT/CATALOG  
 BCB7 ENTRY LENGTH IN DIRECTORY FILE  
 BCB8 ENTRIES PER BLOCK IN DIRECTORY FILE  
 BCB9 FILE COUNT FROM DIRECTORY FILE  
 BCBB DIRECTORY ENTRY NUMBER COUNTER

BCBC \*\*\*\*\* PATHNAME 1 BUFFER \*\*\*\*\*  
 BCBC COMMAND OR PATH LENGTH  
 BCBD TXBUF (COMMAND OR PATHNAME STRING)  
 BCFD NOT USED

BCFE \*\*\*\*\* OPEN FILE NAME TABLE \*\*\*\*\*  
 (EACH ENTRY IS 32 BYTES LONG)  
 (THERE ARE 8 ENTRIES)

BCFE FILE 0: LENGTH OF NAME  
 BCFF FILE 0: L VALUE LSB  
 BD00 FILE 0: L VALUE MSB  
 BD01 FILE 0: START OF NAME STRING  
 (FILE NAME IS STORED BACKWARDS)  
 BD0E LAST 2 BYTES NOT USED

### BASIC INTERPRETER GLOBAL PAGE

This page of memory is rigidly defined by the ProDOS BI. Fields given here will not move in later versions of ProDOS and may be referenced by external, user-written programs. Future additions to the global page may be made in areas which are marked "Not used".

PRODOS BI Global Page		NEXT OBJECT ADDRESS: BE00	
ADDR	LABEL	CONTENTS	
BE00-BE02	BI.ENTRY	JMP to WARMDS (BI warmstart vector).	
BE03-BE05	DOSCMD	JMP to SYNTAX (BI command line parse and execute).	
BE06-BE08	EXTRNCMD	JMP to user-installed external command parser.	
BE09-BE0B	ERROUT	JMP to BI error handler.	
BE0C-BE0E	PRINTERR	JMP to BI error message print routine.	
BE0F	ERRCODE	Place error number in A-register.	
BE10-BE1F	OUTVEC	PRODOS error code (also at \$DE, Applesoft ONERR code).	
BE20-BE2F	INVEC	Default output vector in monitor and for each slot (1-7).	
BE30-BE31	VECTOUT	Default input vector in monitor for each slot (1-7).	
BE32-BE33	VECTIN	Current output vector.	
BE34-BE35	VDOSIO	Current input vector.	
BE36-BE37	VYSIO	BI's output intercept address.	
BE38-BE3B	DEFSLT	BI's input intercept address.	
BE3C	DEFDRV	BI's internal redirection by STATE.	
BE3D	PREGA	Default slot.	
BE3E	PREGX	Default drive.	
BE3F	PREGY	A-register savearea.	
BE40	DTRACE	X-register savearea.	
BE41	STATE	Y-register savearea.	
BE42	EXACTV	Applesoft TRACE is enabled flag (MSB on).	
BE43	IFILACTV	Current intercept state. 0 = immediate command mode. >0 = deferred.	
BE44	OFILACTV	EXEC file active flag (MSB on).	
BE45	PFILACTV	READ file active flag (MSB on).	
BE46	DIRFLG	WRITE file active flag (MSB on).	
BE47	EDIRFLG	PREFIX read active flag (MSB on).	
BE48	STRINGS	File being READ is a DIR file (MSB on).	
BE49	TBUFPTR	End of directory flag (no longer used).	
BE4A	INPTR	String space count used to determine when to garbage collect.	
BE4B	CHRLAST	Buffered WRITE data length.	
BE4C	OPENCNT	Command line assembly length.	
BE4D	YXFILE	Previous output character (for recursion check).	
BE4E	CATFLAG	Number of files open (not counting EXEC).	
BE4F	XTRNADDR	EXEC file being closed flag (MSB on).	
BE50-BE51	XLEN	Line type to format next in DIR file READ.	
BE52		External command handler address.	
		Length of command name (less one).	

ProDOS BI Global Page		NEXT OBJECT ADDRESS: BE53	
ADDR	LABEL	CONTENTS	

BE53 XCNUM      Number of command:  
 \$00 = external      \$0B = OPEN      \$15 = WRITE  
 \$01 = IN#            \$0C = READ      \$16 = APPEND  
 \$02 = PR#            \$0D = SAVE      \$17 = CREATE  
 \$03 = CAT            \$0E = BLOAD     \$18 = DELETE  
 \$04 = FRE            \$0F = BSAVE     \$19 = PREFIX  
 \$05 = BYE            \$10 = CHAIN     \$1A = RENAME  
 \$06 = RUN            \$11 = CLOSE     \$1B = UNLOCK  
 \$07 = BRUN           \$12 = FLUSH     \$1C = VERIFY  
 \$08 = EXEC           \$13 = NOMON     \$1D = CATALOG  
 \$09 = LOAD           \$14 = STORE     \$1E = RESTORE  
 \$0A = LOCK           \$1F = POSITION

BE54-BE55 PBITS      Permitted command operands bits:  
 \$8000      Prefix needed. Pathname optional.  
 \$4000      Slot number only (PR# or IN#).  
 \$2000      Deferred command.  
 \$0800      File name optional.  
 \$0400      If file does not exist, create it.  
 \$0200      T: file type permitted.  
 \$0100      Second file name required.  
 \$0080      First file name required.  
 \$0040      Ab: address keyword permitted.  
 \$0020      B: byte offset permitted.  
 \$0010      E: ending address permitted.  
 \$0008      L: length permitted.  
 \$0004      @: line number permitted.  
 \$0002      S or D: slot/drive permitted.  
 (V always permitted but ignored.)

BE56-BE57 FBITS      Operands found on command line. Same bit  
 assignments as above.  
 VADDR      A keyword value.  
 VBYTE      B keyword value.  
 VENDA      E keyword value.  
 VLNTH      L keyword value.  
 VSLOT      S keyword value.  
 VDRIV      D keyword value.  
 VFELD      F keyword value.  
 VRECD      R keyword value.  
 WVOLM      V keyword value (ignored).  
 VLINE      @ keyword value.  
 VTYPE      T keyword value (in hex).  
 VIOSLT      PR# or IN# slot number value.

ProDOS BI Global Page		NEXT OBJECT ADDRESS: BE6C	
ADDR	LABEL	CONTENTS	

BE6C-BE6D VPATH1      Primary pathname buffer (address of  
 length byte).  
 BE6E-BE6F VPATH2      Secondary pathname buffer (address of  
 length byte).  
 BE70-BE84 GOSYSTEM      Call the MLI using the parameter tables  
 which follow.  
 BE85      MLI call number for this call.  
 BE86-BE87 SYSPARM      Address of MLI parameter list for this  
 call.  
 BE88-BE8A      Return from MLI call.  
 BE8B-BE9E      MLI error return: translate error code to  
 BI error number.  
 BE9F      Not used.  
 BEA0-BEAB      CREATE parameter list.  
 BEAC-BEAE      GET\_PREFIX, SET\_PREFIX, DESTROY parameter  
 list.  
 BEAF-BEB3      RENAME parameter list.  
 BEB4-BEC5      GET\_FILE\_INFO, SET\_FILE\_INFO parameter  
 list.  
 BEC6-BECA      ONLINE, SET MARK, GET MARK, SET\_EOF,  
 GET\_EOF, SET\_BUF, GET\_BUF, QUIT\_parameter  
 list.  
 BECB-BED0      OPEN parameter list.  
 BED1-BED4      SET\_NEWLINE parameter list.  
 BED5-BEDC      READ, WRITE parameter list.  
 BEDD-BEDE      CLOSE, FLUSH parameter list.  
 BEDE-BEF4      "COPYRIGHT APPLE, 1983"  
 BEF5-BEF7      GETBUFR buffer allocation subroutine  
 vector.  
 BEF8-BEFA      FREEBUFR buffer free subroutine vector.  
 BEFB      Original HIMEM MSB.  
 BEFC-BEFF      Not used.

Disk Controller Boot ROM -- Apple II/II+/IIE NEXT OBJECT ADDR: C600  
 ADDR DESCRIPTION/CONTENTS

C600 MODULE STARTING ADDRESS  
 \*\*\*\*\*  
 \* BOOT ROM - APPLE DISK CONTROLLER \*  
 \* FOR APPLE II, II+, AND IIE. \*  
 \* THIS CODE RESIDES FROM \$C600 \*  
 \* TO \$C6FF. IT LOADS TRACK 0 \*  
 \* SECTOR 0 INTO RAM AT \$800 AND \*  
 \* JUMPS TO IT \*  
 \*\*\*\*\*  
 \*\*\*\*\* ZERO PAGE ADDRESSES \*\*\*\*\*

0026 SECTOR BUFFER POINTER  
 002B SLOT NUMBER \* 16 FOR INDEX  
 003C WORKBYTE  
 003D SECTOR WANTED  
 0040 TRACK FOUND  
 0041 TRACK WANTED

\*\*\*\*\* EXTERNAL ADDRESSES \*\*\*\*\*  
 SYSTEM STACK  
 AUXILIARY BUFFER  
 TRANSLATE TABLE  
 SECTORS TO LOAD  
 ENTRY POINT  
 PHASE0 OFF  
 PHASE0 ON  
 MOTOR ON  
 DRIVE SELECT  
 C08A READ DATA REGISTER  
 C08C SET READ MODE  
 C08E MONITOR WAIT ROUTINE  
 FCAB RTS  
 FF58

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

Disk Controller Boot ROM -- Apple II/II+/IIE NEXT OBJECT ADDR: C617  
 ADDR DESCRIPTION/CONTENTS

C617 IF MORE THAN ONE BIT ON, TRY ANOTHER PATTERN >>C614  
 C619 FOUND ONE, GET TABLE VALUE  
 C61A AND STORE IT IN TABLE (0356)  
 C61D INCREMENT TABLE VALUE INDICATOR  
 C61E GET NEXT BIT PATTERN, DONE YET  
 C61F NO, GO CHECK IT OUT >>C606

C621 \*\*\*\*\* DETERMINE SLOT, TURN DRIVE ON \*\*\*\*\*  
 C621 CALL A KNOWN RTS <FF58>  
 C624 GET STACK POINTER  
 C625 GET HIGH BYTE OF WHERE WE ARE (0100)  
 C628 TIMES 16 TO GET SLOT  
 C62C SAVE SLOT  
 C62E PUT IN X REG FOR INDEX  
 C62F INSURE READ MODE (C08E)  
 C635 SELECT DRIVE 1 (C08A)  
 C638 TURN THE MOTOR ON (C089)

C63B \*\*\*\*\* RECALIBRATE DISK ARM \*\*\*\*\*  
 C63B PREPARE TO STEP THE ARM 80 PHASES  
 C63D TURN A PHASE OFF (C080)  
 C640 PUT COUNTER IN ACCUMULATOR  
 C641 CREATE A PHASE NUMBER (0-3)  
 C643 DOUBLE IT FOR PROPER INDEX  
 C644 COMBINE WITH SLOT FOR FINAL INDEX  
 C646 PUT INDEX IN X REGISTER  
 C647 TURN A PHASE ON (C081)  
 C64A DELAY ABOUT 20 MICROSECONDS  
 C64F DECREMENT COUNTER  
 C650 LOOP UNTIL ALL 80 ARE DONE >>C63D

C652 \*\*\*\*\* INITIALIZATION \*\*\*\*\*  
 ---  
 C654 SECTOR TO FIND -> \$00  
 C656 TRACK TO FIND -> \$00  
 C65A MAIN BUFFER POINTER (\$26) -> \$0800  
 C65C CLEAR THE CARRY  
 C65D PUSH STATUS ON STACK

C65E \*\*\*\*\* SEARCH FOR A VALID HEADER \*\*\*\*\*  
 C65E CHECK DATA REGISTER (C08C)  
 C661 LOOP UNTIL DATA IS VALID >>C65E  
 C663 IS IT A \$D5?  
 C665 NO, TRY AGAIN >>C65E  
 C667 YES, CHECK REGISTER AGAIN (C08C)  
 C66A LOOP UNTIL VALID >>C667  
 C66C IS IT AN \$AA

Disk Controller Boot ROM -- Apple II/II+/IIE NEXT OBJECT ADDR: C66E  
 ADDR DESCRIPTION/CONTENTS

C66E NO, SEE IF ITS A \$D5 >>C663  
 C670 YES, DELAY FOR REGISTER TO CLEAR  
 C671 CHECK REGISTER (C08C)  
 C674 LOOP UNTIL VALID >>C671  
 C676 IS IT A \$96  
 C678 YES, WE FOUND AN ADDRESS HEADER >>C683  
 C67A NO, HAVE WE FOUND ONE PREVIOUSLY?  
 C67B IF NOT, START OVER >>C65C  
 C67D WAS IT AN \$AD?  
 C67F YES, WE FOUND A DATA HEADER >>C6A6  
 C681 NO, START OVER >>C65C

C683 \*\*\*\*\* DECODE ADDRESS FIELD \*\*\*\*\*  
 C683 INITIALIZE COUNTER  
 C685 SAVE VALUE DECODED, WILL BE TRACK ON LAST PASS  
 C687 READ DATA REGISTER (C08C)  
 C68A LOOP UNTIL DATA VALID >>C687  
 C68C SHIFT BITS INTO POSITION XIXIXI  
 C68D SAVE FOR LATER  
 C68F READ REGISTER FOR NEXT BYTE (C08C)  
 C692 LOOP UNTIL VALID >>C68F  
 C694 COMBINE WITH PREVIOUS IXIXIX AND XIXIXI  
 C696 DECREMENT COUNTER, DONE YET?  
 C697 NO, DO ANOTHER >>C685  
 C699 KEEP THE STACK CLEAN  
 C69A IS THIS SECTOR WE WANT?  
 C69C NO, START OVER >>C65C  
 C69E GET TRACK FOUND  
 C6A0 IS IT TRACK WE WANT?  
 C6A2 NO, START OVER >>C65C  
 C6A4 YES, INDICATE ADDRESS FOUND, GO LOOK FOR DATA FIELD >>C65D

C6A6 \*\*\*\*\* READ DATA FIELD \*\*\*\*\*  
 C6A6 INITIALIZE OFFSET (AUXILIARY BUFFER)  
 C6A8 ---  
 C6AA READ DATA REGISTER (C08C)  
 C6AD LOOP UNTIL VALID >>C6AA  
 C6AF EXCLUSIVE-OR WITH TRANSLATE TABLE (02D6)  
 C6B4 DECREMENT OFFSET  
 C6B5 STORE BYTE IN AUXILIARY BUFFER (0300)  
 C6B8 LOOP UNTIL BUFFER FULL >>C6A8  
 C6BA INITIALIZE OFFSET (MAIN BUFFER)  
 C6BC READ DATA REGISTER (C08C)  
 C6BF LOOP UNTIL VALID >>C6BC  
 C6C1 EXCLUSIVE-OR WITH TRANSLATE TABLE (02D6)  
 C6C6 STORE BYTE IN MAIN BUFFER  
 C6C8 INCREMENT OFFSET  
 C6C9 LOOP UNTIL BUFFER FULL >>C6BA  
 C6CB READ DATA REGISTER (C08C)

Disk Controller Boot ROM -- Apple II/II+/IIE NEXT OBJECT ADDR: C6CE  
 ADDR DESCRIPTION/CONTENTS

C6CE LOOP UNTIL VALID >>C6CB  
 C6D0 IS CHECKSUM OKAY? (02D6)  
 C6D3 NO, START OVER >>C65C  
 C6D5 \*\*\*\*\* MERGE MAIN AND AUXILIARY BUFFERS \*\*\*\*\*

C6D5 INITIALIZE OFFSET (MAIN BUFFER)  
 C6D7 INITIALIZE OFFSET (AUXILIARY BUFFER)  
 C6D9 DECREMENT OFFSET (AUX BUFFER)  
 C6DA IF LESS THAN ZERO RESET IT >>C6D7  
 C6DC GET BYTE FROM MAIN BUFFER  
 C6E1 ROLL IN TWO BITS FROM AUXILIARY BUFFER  
 C6E6 SAVE COMPLETED DATA BYTE  
 C6E8 INCREMENT OFFSET (MAIN BUFFER)  
 C6E9 LOOP UNTIL WHOLE BUFFER IS DONE >>C6D9

C6EB \*\*\*\*\* DETERMINE IF THERE IS MORE TO DO \*\*\*\*\*  
 C6EB INCREMENT MAIN BUFFER POINTER  
 C6ED INCREMENT SECTOR NUMBER  
 C6F1 IS THERE ANOTHER SECTOR TO LOAD? (0800)  
 C6F6 YES, GO DO IT >>C6D3  
 C6F8 NO, ENTER CODE WE JUST LOADED >>0801

C6FB \*\*\*\*\* UNUSED \*\*\*\*\*  
 C6FB 5 BYTES AT END OF PAGE ARE UNUSED

Disk Controller Boot ROM -- Apple IIC      NEXT OBJECT ADDR: C552  
 ADDR    DESCRIPTION/CONTENTS

Disk Controller Boot ROM -- Apple IIC      NEXT OBJECT ADDR: C552  
 ADDR    DESCRIPTION/CONTENTS

C552    MODULE STARTING ADDRESS

C552 \*\*\*\*\* SLOT5 CODE \*\*\*\*\*

```

*****
*    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 IIC ROM
*    THAT SUPPORTS THE UNIDISK 3.5,
*    26 JULY 85.
*****

```

```

*****
THE FOLLOWING TWO ROUTINES ARE IN THE $C500
AREA BUT ARE USED BY THE $C600 LOGIC.

```

C552 \*\*\*\*\* BOOTFAIL \*\*\*\*\*  
 COME HERE IF BOOT FAILS. PUT MESSAGE ON  
 SCREEN AND GO TO SLEEP FOREVER.

C552 17 CHARACTERS IN MESSAGE  
 C557 PUT AT BOTTOM OF SCREEN (07DB)  
 C55D THEN GO TO SLEEP >>C55D

C55F 'Check Disk Drive'

C56F \*\*\*\*\* SKIP OVER MISCELLANEOUS CODE \*\*\*\*\*

C56F SLOT 5 LOGIC IN HERE

C58E \*\*\*\*\* BUILD READ TRANSLATE TABLE \*\*\*\*\*

```

***** ZERO PAGE ADDRESSES *****
SLOT PAGE PUT HERE DURING AUTOBOOT
RETRY COUNT (HIGH BYTE)
SECTOR BUFFER POINTER
SLOT NUMBER * 16 FOR INDEX
WORKBYTE
SECTOR WANTED
TRACK FOUND
TRACK WANTED
DRIVE TO BOOT FROM

```

```

C58E INITIALIZE BIT PATTERN
C590 INITIALIZE TABLE VALUE INDICATOR
C592 STORE BIT PATTERN
C595 SHIFT PATTERN LEFT ONE BIT
C596 ARE THERE ANY TWO ADJACENT BITS ON?
C598 NO, TRY ANOTHER PATTERN >>C5AA
C59A YES, TURN OFF RIGHTMOST OF EACH GROUP OF ZEROES
C59C FLIP BITS, PAIR OF ZERO BITS NOW SINGLE BIT, ETC
C59E HIGH BIT ALWAYS ON/TURN OFF BIT WE MISSED BEFORE
C5A0 --- >>C5AA

```

\*\*\*\*\* EXTERNAL ADDRESSES \*\*\*\*\*

```

0300 AUXILIARY BUFFER
0356 TRANSLATE TABLE
07DB SCREEN LOCATION
0800 SECTORS TO LOAD
0801 ENTRY POINT
0808 PHASE0 OFF
0811 PHASE0 ON
0888 MOTOR OFF
0889 MOTOR ON
08C8 READ DATA REGISTER
08E8 SET READ MODE
08EA DRIVE SELECT
FC88 MONITOR WAIT ROUTINE

```

```

C5A2 SHIFT PATTERN RIGHT, MUST HAVE ONLY ONE BIT ON
C5A3 IF MORE THAN ONE BIT ON, TRY ANOTHER PATTERN >>C5A0
C5A5 FOUND ONE, GET TABLE VALUE
C5A6 AND STORE IT IN TABLE (0356)
C5A9 INCREMENT TABLE VALUE INDICATOR
C5AA GET NEXT BIT PATTERN, DONE YET?
C5AB NO, GO CHECK IT OUT >>C592
C5AD MAIN BUFFER POINTER ($26) --> $0800
C5B1 INITIALIZE RETRY COUNT (LOW BYTE)
C5B3 RETURN TO CALLER

```

C5B4 \*\*\*\*\* SKIP OVER MISCELLANEOUS CODE \*\*\*\*\*

C5B4 SLOT 5 LOGIC IN HERE



Disk Controller Boot ROM -- Apple Iic NEXT OBJECT ADDR: C5F5

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

C5F5 \*\*\*\*\* JUMP TO BOOTFAIL \*\*\*\*\*  
 C5F5 BRANCH TO BOOTFAIL >>C552  
 C5F8 REMAINING 8 BYTES NOT USED BY DISK II >>C576  
 C600 \*\*\*\*\* INITIALIZATION \*\*\*\*\*

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

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

---  
 C608 INITIALIZE SLOT (6)  
 C60D INITIALIZE DEVICE (1 OR 2)  
 C60F SAVE DRIVE NUMBER ON STACK  
 C610 INSURE READ MODE (C08E)  
 C616 GET DRIVE NUMBER BACK  
 C617 SELECT APPROPRIATE DRIVE (C0EA)  
 C61A TURN MOTOR ON (C089)

C61D \*\*\*\*\* RECALIBRATE DISK ARM \*\*\*\*\*  
 C61D PREPAIR TO STEP THE ARM 80 PHASES  
 C61F TURN A PHASE OFF (C080)  
 C622 PUT COUNTER IN A REGISTER  
 C623 CREATE A PHASE NUMBER (0-3)  
 C625 DOUBLE IT FOR PROPER INDEX  
 C626 COMBINE WITH SLOT FOR FINAL INDEX  
 C628 PUT INDEX IN X REGISTER  
 C629 TURN A PHASE ON (C081)  
 C62C DELAY ABOUT 20 MICROSECONDS  
 C631 DECREMENT COUNTER  
 C632 LOOP UNTIL ALL 80 ARE DONE >>C61F

C634 \*\*\*\*\* INITIALIZATION \*\*\*\*\*  
 ---  
 C636 SECTOR TO FIND -> \$00  
 C638 TRACK TO FIND -> \$00  
 C63A BUILD THE TRANSLATE TABLE <C58E>

C63D \*\*\*\*\* COUNT RETRIES AND INDICATE ERROR IF BOOT FAILS \*\*\*\*\*  
 C63D INITIALIZE RETRY COUNT  
 C63F CLEAR THE CARRY  
 C640 PUSH STATUS ON STACK  
 C641 KEEP STACK CLEAN  
 C642 GET SLOT

Disk Controller Boot ROM -- Apple Iic NEXT OBJECT ADDR: C644

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

C644 DECREMENT RETRY COUNT, TRY AGAIN?  
 C646 YES, GO DO IT >>C656  
 C648 NO, TURN DRIVE OFF (C088)  
 C64B AUTO BOOT FROM SLOT6?  
 C64F NO, FAIL NOW >>C5F5  
 C651 MAYBE SLOT 5 WILL TALK TO US >>C500  
 C654 TWO BYTES NOT USED >>0002  
 ---  
 C657 DECREMENT RETRY COUNT (LOW BYTE)  
 C658 IF NOT ZERO, TRY AGAIN >>C65E  
 C65A IF SO, GO DECREMENT RETRY COUNT (HIGH BYTE) >>C641  
 C65C SPACE FILLER TO POSITION CODE BELOW >>C63D

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

---  
 C65E CHECK DATA REGISTER (C08C)  
 C661 LOOP UNTIL DATA IS VALID >>C65E  
 C663 IS IT A \$D5?  
 C665 NO, TRY AGAIN >>C657  
 C667 YES, CHECK REGISTER AGAIN (C08C)  
 C66A LOOP UNTIL VALID >>C667  
 C66C IS IT AN SAA  
 C66E NO, SEE IF ITS A \$D5 >>C663  
 C670 YES, DELAY FOR REGISTER TO CLEAR  
 C671 CHECK REGISTER (C08C)  
 C674 LOOP UNTIL VALID >>C671  
 C676 IS IT A \$96  
 C678 YES, WE FOUND AN ADDRESS HEADER >>C683  
 C67A NO, HAVE WE FOUND ONE PREVIOUSLY?  
 C67B IF NOT, START OVER >>C63F  
 C67D WAS IT AN \$AD?  
 C67F YES, WE FOUND A DATA HEADER >>C6A6  
 C681 NO, START OVER >>C63F

C683 \*\*\*\*\* DECODE ADDRESS FIELD \*\*\*\*\*

C683 INITIALIZE COUNTER  
 C685 SAVE VALUE DECODED, WILL BE TRACK ON LAST PASS  
 C687 READ DATA REGISTER (C08C)  
 C68A LOOP UNTIL DATA VALID >>C687  
 C68C SHIFT BITS INTO POSITION XLXIXI  
 C68D SAVE FOR LATER  
 C68F READ REGISTER FOR NEXT BYTE (C08C)  
 C692 LOOP UNTIL VALID >>C68F  
 C694 COMBINE WITH PREVIOUS IXLXIXI AND XLXIXI  
 C696 DECREMENT COUNTER, DONE YET?  
 C697 NO, DO ANOTHER >>C685  
 C699 KEEP THE STACK CLEAN  
 C69A IS THIS SECTOR WE WANT?  
 C69C NO, START OVER >>C63F  
 C69E GET TRACK FOUND

Disk Controller Boot ROM -- Apple IIc                   NEXT OBJECT ADDR: C6A0  
 -----  
 ADDR   DESCRIPTION/CONTENTS  
 -----

```

C6A0 IS IT TRACK WE WANT?
C6A2 NO, START OVER >>C63F
C6A4 YES, INDICATE ADDRESS FOUND, GO LOOK FOR DATA FIELD >>C642

C6A6 ***** READ DATA FIELD *****
C6A6 INITIALIZE OFFSET (AUXILIARY BUFFER)
C6A8 ---
C6AA READ DATA REGISTER (C08C)
C6AD LOOP UNTIL VALID >>C6AA
C6AF EXCLUSIVE-OR WITH TRANSLATE TABLE (02D6)
C6B4 DECREMENT OFFSET
C6B5 STORE BYTE IN AUXILIARY BUFFER (0300)
C6B8 LOOP UNTIL BUFFER FULL >>C6A8
C6BA INITIALIZE OFFSET (MAIN BUFFER)
C6BC READ DATA REGISTER (C08C)
C6BF LOOP UNTIL VALID >>C6BC
C6C1 EXCLUSIVE-OR WITH TRANSLATE TABLE (02D6)
C6C6 STORE BYTE IN MAIN BUFFER
C6C8 INCREMENT OFFSET
C6C9 LOOP UNTIL BUFFER FULL >>C6BA
C6CB READ DATA REGISTER (C08C)
C6CE LOOP UNTIL VALID >>C6CB
C6D0 IS CHECKSUM OKAY? (02D6)
C6D3 NO, START OVER >>C6A2

C6D5 ***** MERGE MAIN AND AUXILIARY BUFFERS*****
C6D5 INITIALIZE OFFSET (MAIN BUFFER)
C6D7 INITIALIZE OFFSET (AUXILIARY BUFFER)
C6D9 DECREMENT OFFSET (AUX BUFFER)
C6DA IF LESS THAN ZERO RESET IT >>C6D7
C6DC GET BYTE FROM MAIN BUFFER
C6E1 ROLL IN TWO BITS FROM AUXILIARY BUFFER
C6E6 SAVE COMPLETED DATA BYTE
C6E8 INCREMENT OFFSET (MAIN BUFFER)
C6E9 LOOP UNTIL WHOLE BUFFER IS DONE >>C6D9

C6EB ***** DETERMINE IF THERE IS MORE TO DO*****
C6EB INCREMENT MAIN BUFFER POINTER
C6ED INCREMENT SECTOR NUMBER
C6F1 IS THERE ANOTHER SECTOR TO LOAD? (0800)
C6F6 YES, GO DO IT >>C6D3
C6F8 NO, ENTER CODE WE JUST LOADED >>0801

C6FB 5 ZERO BYTES AT END OF PAGE

```

Disk II Boot ROM -- Apple IIGS  
 ADDR DESCRIPTION/CONTENTS  
 -----

Disk II Boot ROM -- Apple IIGS  
 ADDR DESCRIPTION/CONTENTS  
 -----

Next Object Addr: C600  
 Next Object Addr: C600

C600 MODULE STARTING ADDRESS  
 \*\*\*\*\*  
 \*  
 \* C600 ROM - APPLE IIGS  
 \* If an Apple IIGS has slot 6  
 \* configured for Disk Port,  
 \* then this code is executed  
 \* when a boot is attempted  
 \* from slot 6.  
 \*  
 \* IIGS ROM VERSION 0  
 \*  
 \*\*\*\*\*

C600 \*\*\*\*\* IIGS SOFT SWITCHES \*\*\*\*\*  
 C02D SLTROMSEL. SLOT ROM CONFIGURATION BYTE  
 C035 SHADOW. ENABLES/DISABLES SHADOWING  
 C068 STATEREG. ONE BYTE SETS 8 SOFT SWITCHES  
 C600 \*\*\*\*\* C600 ENTRY POINT \*\*\*\*\*

C600 \*\*\*\*\* ZERO PAGE ADDRESSES \*\*\*\*\*  
 1 SLOT PAGE PUT HERE DURING AUTOBOOT  
 3 RETRY COUNT (HIGH BYTE)  
 26 SECTOR BUFFER POINTER  
 2B SLOT NUMBER \* 16  
 3C WORKBYTE  
 3D SECTOR WANTED  
 40 TRACK FOUND  
 41 TRACK WANTED  
 4F DRIVE TO BOOT FROM

C622 \*\*\*\*\* COUNT RETRIES AND INDICATE ERROR \*\*\*\*\*  
 IF BOOT FAILS

C600 \*\*\*\*\* EXTERNAL ADDRESSES \*\*\*\*\*  
 0300 AUXILIARY BUFFER  
 0356 TRANSLATE TABLE  
 07F8 SLOT THAT OWNS C800-CFFF  
 07FE UTILITY BYTE FOR SLOT 6  
 0800 SECTORS TO LOAD  
 0801 ENTRY POINT  
 C080 PHASE0 OFF  
 C081 PHASE0 ON  
 C088 MOTOR OFF  
 C089 MOTOR ON  
 C08C READ DATA REGISTER  
 C08E SET READ MODE  
 C0EA DRIVE SELECT  
 E000 APPLESOFT BASIC ENTRY POINT  
 FABA MONITOR DISK CONTROLLER SEARCH LOOP  
 FC48 MONITOR WAIT ROUTINE

C622 \*\*\*\*\* SUBROUTINE TO ENABLE INTERRUPTS \*\*\*\*\*  
 C622 INITIALIZE RETRY COUNT  
 C624 DISABLE INTERRUPTS (AGAIN)  
 C625 CLEAR CARRY AND  
 C626 PUT IT ON THE STACK.  
 C627 KEEP STACK EVEN  
 C628 GET SLOT\*16 IN X-REGISTER  
 C62A DECREMENT RETRY COUNT. TRY AGAIN?  
 C62C YES, GO DO IT >>C636  
 C62E NO, TURN DRIVE OFF (C088)  
 C631 CALL BOOT ERROR HANDLER <FF5971>  
 C635 ENABLE INTERRUPTS IF OK TO <C648>  
 C638 AUTOBOOT FROM SLOT 6?  
 C63D NO, GO TO BASIC >>C642  
 C63F YES, RETURN TO SLOT SEARCH LOOP >>FABA  
 C642 JUMP TO BASIC >>E000

C645 \*\*\*\*\* SUBROUTINE TO ENABLE INTERRUPTS \*\*\*\*\*  
 C645 ALLOW SHADOWING AND I/O (C035)  
 C648 CHECK ORIGINAL P-REGISTER (07FE)  
 C64B INTERRUPT BIT HIGH?  
 C64D YES, LEAVE INTERRUPTS DISABLED >>C650  
 C64F NO, ENABLE INTERRUPTS  
 C650 RESTORE SECTOR TO ACCUMULATOR  
 C652 RETURN

C645 \*\*\*\*\* SUBROUTINE TO ENABLE INTERRUPTS \*\*\*\*\*  
 C645 ALLOW SHADOWING AND I/O (C035)  
 C648 CHECK ORIGINAL P-REGISTER (07FE)  
 C64B INTERRUPT BIT HIGH?  
 C64D YES, LEAVE INTERRUPTS DISABLED >>C650  
 C64F NO, ENABLE INTERRUPTS  
 C650 RESTORE SECTOR TO ACCUMULATOR  
 C652 RETURN

Disk II Boot ROM -- Apple IIGS  
 ADDR DESCRIPTION/CONTENTS  
 -----  
 NEXT OBJECT ADDR: C652

C653 \*\*\*\*\* THREE BYTES NOT USED \*\*\*\*\*  
 C653 NOT USED  
 C656 \*\*\*\*\* INCREMENT RETRIES \*\*\*\*\*  
 ---  
 C657 DECREMENT RETRY COUNT (LOW BYTE)  
 C658 IF NOT ZERO, TRY AGAIN >>C65E  
 C65A ZERO, SO GO DECREMENT HIGH BYTE >>C627  
 C65C SPACE FILLER TO POSITION CODE BELOW >>C622  
 C65E \*\*\*\*\* SEARCH FOR A VALID HEADER \*\*\*\*\*  
 C65E CHECK DATA REGISTER (C08C)  
 C661 LOOP UNTIL DATA IS VALID >>C65E  
 C663 IS IT A \$D5?  
 C665 NO, TRY AGAIN >>C657  
 C667 YES, CHECK REGISTER AGAIN (C08C)  
 C66A LOOP UNTIL VALID >>C667  
 C66C IS IT AN \$AA?  
 C66E NO, SEE IF IT'S A \$D5 >>C663  
 C670 YES, DELAY FOR REGISTER TO CLEAR  
 C671 CHECK REGISTER (C08C)  
 C674 LOOP UNTIL VALID >>C671  
 C676 IS IT A \$96?  
 C678 YES, WE FOUND AN ADDRESS HEADER >>C683  
 C67A NO. ARE WE LOOKING FOR DATA HEADER?  
 C67B NO, SO START OVER >>C625  
 C67D YES, WAS IT AN \$SAD?  
 C67F YES, WE FOUND A DATA HEADER >>C6A6  
 C681 NO, START OVER >>C625

C683 \*\*\*\*\* DECODE ADDRESS FIELD \*\*\*\*\*  
 C683 INITIALIZE COUNTER  
 C685 SAVE VALUE DECODED, WILL BE TRACK ON LAST PASS  
 C687 READ DATA REGISTER (C08C)  
 C68A LOOP UNTIL DATA VALID >>C687  
 C68C SHIFT BITS INTO POSITION X1X1X1  
 C68D SAVE FOR LATER  
 C68F READ REGISTER FOR NEXT BYTE (C08C)  
 C692 LOOP UNTIL VALID >>C68F  
 C694 COMBINE WITH PREVIOUS 1X1X1X1X AND X1X1X1X1  
 C696 DECREMENT COUNTER. DONE YET?  
 C697 NO, DO ANOTHER >>C685  
 C699 KEEP STACK CLEAN  
 C69A IS THIS SECTOR WE WANT?  
 C69C NO, START OVER >>C625  
 C69E GET TRACK FOUND

Disk II Boot ROM -- Apple IIGS  
 ADDR DESCRIPTION/CONTENTS  
 -----  
 NEXT OBJECT ADDR: C6A0

C6A0 IS IT TRACK WE WANT?  
 C6A2 NO, START OVER >>C625  
 C6A4 YES, INDICATE ADDR FOUND, GO LOOK FOR DATA FIELD >>C628  
 C6A6 \*\*\*\*\* READ DATA FIELD \*\*\*\*\*  
 C6A6 INITIALIZE OFFSET (AUXILIARY BUFFER)  
 C6A8 ---  
 C6AA READ DATA REGISTER (C08C)  
 C6AD LOOP UNTIL VALID >>C6AA  
 C6AF EXCLUSIVE-OR WITH TRANSLATE TABLE (02D6)  
 C6B4 DECREMENT OFFSET  
 C6B5 STORE BYTE IN AUXILIARY BUFFER (0300)  
 C6B8 LOOP UNTIL BUFFER FULL >>C6A8  
 C6BA INITIALIZE OFFSET (MAIN BUFFER)  
 C6BC READ DATA REGISTER (C08C)  
 C6BF LOOP UNTIL VALID >>C6BC  
 C6C1 EXCLUSIVE-OR WITH TRANSLATE TABLE (02D6)  
 C6C6 STORE BYTE IN MAIN BUFFER  
 C6C8 INCREMENT OFFSET  
 C6C9 LOOP UNTIL BUFFER FULL >>C6BA  
 C6CB READ DATA REGISTER (C08C)  
 C6CE LOOP UNTIL VALID >>C6CB  
 C6D0 IS CHECKSUM OKAY? (02D6)  
 C6D3 NO, START OVER >>C6A2  
 C6D5 \*\*\*\*\* MERGE MAIN AND AUXILIARY BUFFERS \*\*\*\*\*  
 C6D5 INITIALIZE OFFSET (MAIN BUFFER)  
 C6D7 INITIALIZE OFFSET (AUXILIARY BUFFER)  
 C6D9 DECREMENT OFFSET (AUX BUFFER)  
 C6DA IF LESS THAN ZERO RESET IT >>C6D7  
 C6DC GET BYTE FROM MAIN BUFFER  
 C6E1 ROLL IN TWO BITS FROM AUXILIARY BUFFER  
 C6E6 SAVE COMPLETED DATA BYTE  
 C6E8 INCREMENT OFFSET (MAIN BUFFER)  
 C6E9 LOOP UNTIL WHOLE BUFFER IS DONE >>C6D9  
 C6EB \*\*\*\*\* DETERMINE IF THERE IS MORE TO DO \*\*\*\*\*  
 C6EB INCREMENT MAIN BUFFER POINTER  
 C6ED INCREMENT SECTOR NUMBER  
 C6F1 IS THERE ANOTHER SECTOR TO LOAD? (0800)  
 C6F6 YES, GO DO IT >>C6D3  
 C6F8 NO. ENABLE INTERRUPTS <C645>  
 C6FB AND JUMP TO CODE WE JUST LOADED. >>0801

Disk II Boot ROM -- Apple IIGS NEXT OBJECT ADDR: C6FB

```

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

```

```

C6FE LAST TWO BYTES ARE ZERO

*****
*          SUBROUTINES ELSEWHERE IN ROM          *
*          (BANK $FF -- IIGS ROM)                *
*          *****                               *

```

```

5909 ***** SELECT DRIVE AND TURN IT ON *****
-----
590C INITIALIZE SLOT TO 6
590E INITIALIZE DEVICE (1 OR 2)
5910 INSURE READ MODE (C08E)
5916 SELECT APPROPRIATE DRIVE (C0EA)
5919 TURN MOTOR ON (C089)

591C ***** RECALIBRATE DISK ARM *****

```

```

591C PREPARE TO STEP THE ARM 80 PHASES
591E TURN A PHASE OFF (C080)
5921 PUT COUNTER IN A-REG
5922 CREATE A PHASE NUMBER (0-3)
5924 DOUBLE IT FOR PROPER INDEX
5925 COMBINE WITH SLOT FOR FINAL INDEX
5927 PUT INDEX IN X REGISTER
5928 TURN A PHASE OFF (C081)
592B DELAY ABOUT 20 MICROSECONDS
592D SAVE P-REGISTER
592E AND DISABLE INTERRUPTS
592F WHILE WAITING. <FCAB>
5932 RESTORE P-REGISTER
5933 DECREMENT COUNTER
5934 LOOP UNTIL ALL 80 ARE DONE >>591E

```

```

5936 ***** SET VARIABLES TO ZERO *****
5938 ZERO LOW BYTE, BUFFER ADDRESS
593A TRACK = 0
593C CLEAR CARRY, INDICATING READABLE DISK
593D TRY 50 READS
593F READ DATA REGISTER (C08C)
5942 FOUND SOMETHING! >>5948
5944 DECREMENT COUNTER
5945 AND TRY AGAIN. >>593F
5947 CAN'T FIND ANY DATA. SET ERROR INDICATOR.

```

Disk II Boot ROM -- Apple IIGS NEXT OBJECT ADDR: 5947

```

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

```

```

5948 ***** BUILD READ TRANSLATE TABLE *****
5948 SAVE P-REGISTER
5949 INITIALIZE BIT PATTERN
594B INITIALIZE TABLE VALUE INDICATOR
594D STORE BIT PATTERN
5950 SHIFT PATTERN LEFT ONE BIT
5951 ARE THERE ANY TWO ADJACENT BITS ON?
5953 NO. TRY ANOTHER PATTERN >>5966
5955 YES, TURN OFF RIGHTMOST OF EACH GROUP OF ZEROS.
5957 FLIP BITS, PAIR OF ZERO BITS NOW SINGLE BIT, ETC.
5959 HIGH BIT ALWAYS ON/TURN OFF BIT WE MISSED BEFORE
595B --- >>5966

```

```

595D SHIFT PATTERN RIGHT, MUST HAVE ONLY ONE BIT ON
595E IF MORE THAN ONE BIT ON, TRY ANOTHER PATTERN >>595B
5960 FOUND ONE, GET TABLE VALUE
5961 AND STORE IT IN TABLE. (000356)
5965 INCREMENT TABLE VALUE INDICATOR
5966 GET NEXT BIT PATTERN. DONE YET?
5967 NO, GO CHECK IT OUT >>594D
5969 MAIN BUFFER POINTER ($26) -> $800
596D INITIALIZE RETRY COUNT
596F RESTORE P-REGISTER
5970 BACK TO $C600 CODE

```

```

5971 ***** HANDLE BOOT ERROR *****
5971 SET "NOT A STARTUP DISK" ERROR
5977 JUMP INTO BOOT ERROR LOGIC >>63D6

```

```

63D6 ***** BOOT ERROR LOGIC *****
63D6 GET INTO NATIVE MODE
63D8 SAVE P-REGISTER ON STACK
63D9 8-BIT MEMORY AND INDEX OPERATION
63DB AUTO BOOT FROM THIS SLOT?
63E1 YES, DON'T PRINT ERROR MESSAGE >>6423
63E3 SET DATA BANK REGISTER TO BANK 0 <6307>
63E6 SAVE SLOTTROM STATUS (C02D)
63EA ENABLE SMARTPORT/DISKPORT FOR SLOTS 5 AND 6
63EF SAVE STATEREG (C068)
63F3 READ ROM
63F8 GET INTO EMULATION MODE
63FB SET AND CLEAR NORMAL SCREEN IN PREPARATION <06C593>
6400 RESTORE EMULATION MODE
6402 RESTORE SOFT SWITCHES (C068)
6406 RESTORE SLOTTROM STATUS (C02D)

```

Disk II Boot ROM -- Apple IIGS NEXT OBJECT ADDR: 6409

```

-----
ADDR  DESCRIPTION/CONTENTS
-----
6409 SET MSG CODE=0 ("I/O ERROR")
640B IS ERROR CODE ZERO? (E10FB1)
640F NO, KEEP CHECKING >>6413
6411 YES, SET MSG CODE=1 ("NOT A STARTUP DISK")
6413 IS IT "NO DEVICE CONNECTED"?
6415 NO. >>6419
6417 YES, SET MSG CODE=2
6419 IS IT "CHECK STARTUP DEVICE"?
641B NO, CALL IT "I/O ERROR" >>641F
641D YES, SET MSG CODE=3
6420 AND PRINT IT OUT <C000>
6423 INDICATE I/O ERROR
6425 GET P-REGISTER BACK
6426 RESTORE MODE BASED ON CARRY
6427 INDICATE ERROR
6428 RETURN TO $C600 LOGIC

```

---

**APPENDIX A****DIFFERENCES BETWEEN THE PRODOS 8 VERSIONS**

This Appendix identifies the changes to ProDOS 8 that were introduced with the 1.2 and 1.3 versions. Although we believe this to be a fairly thorough list, there may have been a few changes (especially deletions) that we didn't catch. Quite a few changes that were apparently made only to save space are not listed here.

**CHANGES INTRODUCED IN THE 1.2 VERSION**

The changes that were made to the 1.1.1 version of ProDOS to produce ProDOS 8, Version 1.2, are listed below. Addresses given here are Version 1.2 addresses.

**Relocator**

1. When running on a IIGS with ProDOS 16 installed, ProDOS 8 is entered from PQUIT (the ProDOS 16 Quit Handler). In this case the Relocator is entered at \$2003 instead of \$2000. [2000-2005, page 11].
2. The ProDOS 8 version number is now stored in the MLI subdirectory header data area [2048-204C, page 11].
3. Always checks to see if running on a IIGS, and if so sets a flag (2278). Also sets E100BD=0 if ProDOS 8 is the initial boot on a IIGS. [2079-208D, page 11].
4. Sets aux stack pointer to \$FF [20E6-20F5, 2111-2115, page 12].
5. If operating on a IIGS, skips logic that searches for slot 3 80-column card [212A-2139, page 12].
6. If operating on a IIGS, installs the IIGS Clock Code [21AD-21D0, 22D3-22DA, pages 12 and 14].
7. Now checks for an AppleTalk Initialization File (ATINIT file) before looking for a .SYSTEM file. If the ATINIT file is found, it is loaded and executed, then the search for a .SYSTEM file commences. [22DB-2381, page 14].
8. The list of devices is now ordered differently. It recognizes the SmartPort and allows four Slot 5 SmartPort units to be accessed as Slot 5, Drives 1 and 2 and Slot 2, Drives 1 and 2. If Slot 2 is being used by a storage device, however, only the first two devices on the Slot 5 SmartPort can be accessed. It also changes the search order, making sure that Disk II devices are searched last when a device scan takes place (such as during an MLI ON LINE call). [2668-271B, 275D-2767, 2814-28AA, pages 17-19].

9. A bug in the /RAM driver (which we pointed out in the 1.1.1 supplement) that allowed a block read of block 7 (which doesn't exist) has been corrected [2D4B, page 23].
10. The /RAM caller, which operates in high RAM, now contains a \$60 (RTS instruction) at address \$FF58. Peripheral cards sometimes call that address to figure out which slot they are in, and in case they forget to set ROM for reading, the call will still work. [2E56-2E58, page 24].
11. A subroutine that sets high RAM for reading/writing was created to save space in the code [2518-251E, page 16].

### MLI and MLI Global Page

1. The Global Page now pushes the P-Register and disables interrupts before calling the actual MLI [DE01-DE04, DE1C-DE21, BF4B-BF4F, pages 34 and 75].
2. Setting the MLIACTIVE flag a little differently now allows nested calls to the MLI by interrupt routines [DE8F-DE91, page 35].
3. A new MLI command was introduced (command=\$82), that allows the user to install a routine to handle unclaimed interrupts [DEFF-DF0B, FD23-FD2C, pages 35 and 63].
4. If there is no unclaimed interrupt handler, ProDOS 8 now counts unclaimed interrupts, and will allow 255 of them to occur before finally issuing a fatal error. This allows a brief time for the unknown interrupt to stop interrupting. [DFB3-DFB7, page 36].
5. If operating on a IIGS and system death occurs, the NEWVIDEO softswitch is set to 0, reinitializing the IIGS video [E013-E016, page 36].
6. Processing for the ON\_LINE command now frees the VCB entry for a device that was previously on-line but has been taken off-line [E28A-E2A4, page 39].
7. A subroutine that reads a block where the block number is in the A and X registers was added to save space in the code [EBC9-EBD0, page 47].
8. The error message that results when a file being opened has an illegal storage type has been changed from "incompatible directory format" to "unsupported storage type" [EEC7-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. [F992-F99A, FBC7-FBDC, pages 67 and 69].
2. Previous versions of ProDOS forgot, in certain cases, to rewrite index blocks that were being discarded when shortening or deleting files. Now such blocks will always be rewritten to disk in a zeroed (shortened file) or flipped (deleted file) form. [FAB8-FABC, FB91-FB93, pages 68 and 69].
3. A poorly-written loop in the QuitCode Caller that was added for Version 1.2 was rewritten for Version 1.3. The Version 1.2 code might cause problems on a IIGS. [FD05-FD0C, page 70].

**Disk II Device Driver**

1. There is a routine in the Disk II Device Driver that clears phases in case the Disk II device is sharing transmission lines with SmartPort devices. This routine was patched in Version 1.3 so that phases are now cleared with LDA instructions instead of STA instructions. This eliminates bus fights that can, in some situations, cause unwanted writing to the floppy disk. [D6C3-D6CE, page 88].

**BUGS IN VERSIONS 1.2 AND 1.3**

It is fair to say that both Versions 1.2 and 1.3 of ProDOS 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
ORA    BITMAP,Y   PUT BIT PATTERN IN ACCUM
STA    BITMAP,Y   MARK THIS PAGE AS IN USE
...     UPDATE MAP
...     WE'VE GOT IT NOW

```

```

LOCATE PHA          SAVE PAGE NUMBER
      AND #07       ISOLATE BIT POSITION
      TAY          THIS IS INDEX INTO MASK TABLE
      LDX BITMASK,Y PUT PROPER BIT PATTERN IN X
      PLA          RESTORE PAGE NUMBER
      LSR A        DIVIDE PAGE BY 8
      LSR A
      LSR A
      TAY          Y-REG IS OFFSET INTO BITMAP
      TXA          PUT BIT PATTERN IN ACCUM
      RTS          DONE

BITMASK DFB $80,$40,$20,$10 BIT MASK PATTERNS
      DFB $08,$04,$02,$01

```

## Page 7-9

The code on page 7-9 is incorrect and should be replaced with the following:

```

*      SQUISH OUT DEVICE NUMBER FROM DEVLST
      SKP 1
      LDX $BF31      GET DEVCNT
DEVLP  LDA $BF32,X   PICK UP LAST DEVICE NUM
      AND #$70       ISOLATE SLOT
      CMP #$30       SLOT = 3?
      BEQ GOTSLT     YES, CONTINUE
      DEX
      BPL DEVLPL     CONTINUE SEARCH BACKWARDS
      BMI NORAM      CAN'T FIND IT IN DEVLST
GOTSLT LDA $BF32+1,X GET NEXT NUMBER
      STA $BF32,X    AND MOVE THEM FORWARD
      INX
      CPX $BF31      REACHED LAST ENTRY?
      BNE GOTSLT     NO, LOOP
      DEC $BF31      REDUCE DEVCNT BY 1
      LDA #0         ZERO LAST ENTRY IN TABLE
      STA $BF32,X
      CLC
      BCC OKXIT      BRANCH ALWAYS TAKEN
      SKP 1
OLDVEC DW 0         OLD VECTOR SAVEAREA

```

To reinstall the /RAM driver, execute this subroutine:

```

*      SKP      1
      SEE IF SLOT 3 HAS A DRIVER ALREADY
*      SKP      1
HIMEM  EQU      $73          PTR TO BI'S GENERAL PURPOSE BUFFER
      SKP      1
INSTALL LDX     $BF31       GET DEVCNT
INSLP  LDA     $BF32,X     GET A DEVNUM
      AND     #$70        ISOLATE SLOT
      CMP     #$30        SLOT 3?
      BEQ     INSOUT      YES, SKIP IT
      DEX
      BPL     INSLP       KEEP UP THE SEARCH
*      SKP      1
      RESTORE THE DEVNUM TO THE LST
      SKP      1
      LDX     $BF31       GET DEVCNT AGAIN
      CPX     #$0D        DEVICE TABLE FULL?
      BNE     INSLP2
ERROR  ...                YOUR ERROR ROUTINE

INSLP2 LDA     $BF32-1,X   MOVE ALL ENTRIES DOWN
      STA     $BF32,X     TO MAKE ROOM AT FRONT
      DEX
      BNE     INSLP2
      LDA     #$B0
      STA     $BF32       SLOT 3, DRIVE 2 AT TOP OF LIST
      INC     $BF31       UPDATE DEVCNT
      SKP     1

```

#### Page 7-26:

Modifying the ProDOS Disk II Device Driver to allow 320 blocks instead of the normal 280. The fourth command line should read:

```
520D:40
```

Modifying FILER to format 40 tracks instead of 35. The fourth command line should read:

```
4244:40
```

[See Second printing errata for information about versions other than 1.0.1]

**Page 8-6:**

Under "Device Information", make the following changes:

BF10-BF11	DEVADR01	Slot 0 reserved.
...		
BF26-BF27	DEVADR32	/RAM device driver address (need extra 64K).

**Page 8-7:**

The wrong bit is indicated as the "expansion bit" in the MACHID byte. The first eight rows of that description should read:

00.. 0...	II
01.. 0...	II+
10.. 0...	IIE
11.. 0...	III emulation
00.. 1...	Future expansion
01.. 1...	Future expansion
10.. 1...	IIC
11.. 1...	Future expansion

**Page B-8:**

In the last paragraph, the sentence should read "A second way to use an **interpreted** language..." (not a **compiled** language).

**Page D-1:**

In the second paragraph, the sentence should read "Versions of the Disk Drive Controller Unit are now **used**..." (not **based**).

**Reference Card, Panel 4**

Under "SYSTEM GLOBAL PAGE FORMAT", replace the lines beginning BF05 and BF06 with the following two lines:

BF06	Jump to Date/Time Address (or RTS if no clock)
------	---

The description of BF10-11 should be changed to:

BF10-11 Slot 0 reserved

The description of BF26-27 should be changed to:

BF26-27 /RAM

Under the "MACHINE IDENTIFICATION BYTE", the second column of numbers should read:

0...  
0...  
0...  
0...  
1...  
1...  
1...  
1...

#### Reference Card, Panel 9

The last entry for "MLI ERROR CODES" should be:

\$5A Bad vol. bit map

(not \$58).

**ERRATA TO BENEATH APPLE PRODOS (2nd Printing, 1985)**

**Page 4-30:**

The definitions of PARENT POINTER and PARENT ENTRY are incorrect. Replace them with:

\$27-\$28 PARENT\_POINTER: The block number (within the volume directory or a subdirectory) which contains the file entry for this subdirectory.

\$29 PARENT\_ENTRY: The number of the file entry within the block number pointed to by the PARENT\_POINTER. Given that "ENTRIES\_PER\_BLOCK" is \$0D, then the PARENT\_ENTRY number ranges from \$01 to \$0D.

**Page 6-62:**

The paragraph immediately preceding Table 6.6 should read as follows:

If an error occurs, the BI error code will be placed in the accumulator. Possible codes are listed in Table 6.6.

In Table 6.6, the message for error code \$0C is wrong. It should read:

\$0C NO BUFFERS AVAILABLE

**Page 7-26:**

Expand the 40-track drive patch to show how to patch all of the versions of ProDOS 8 released to date.

This patch modifies the Disk II Driver, which is a part of the "PRODOS" file (or "P8" file), so that it allows 320 blocks per volume instead of 280 blocks per volume. First set the prefix to the directory that contains the file you want to modify. This file will normally be called "PRODOS" on an 8-bit Apple II and "P8" on a 16-bit Apple IIGS. If the file name is not "PRODOS," substitute the correct filename wherever "PRODOS" appears.

```
UNLOCK PRODOS
BLOAD PRODOS,TSYS,A$2000
CALL -151
address*:40
3D0G
BSAVE PRODOS,TSYS,A$2000
LOCK PRODOS
```



\*\*"address" varies with the version of ProDOS, as follows:

ProDOS Version	address
1.0.1	520D
1.0.2	52CD
1.1.1	56E3
1.2	58E3
1.3	58E3

The following patch modifies the program FILER to format 40 tracks instead of 35. After this modification is made, only 40-track drives may be formatted with FILER.

```
UNLOCK FILER
BLOAD FILER,TSYS,A$2000
CALL -151
addr**:40
79F4:28
3D0G
BSAVE FILER,TSYS,A$2000
LOCK FILER
```

\*\*"addr" depends on the release date of FILER. Here are the values of "addr" for two different release dates:

Release date	addr
1 JAN 84	4244
18 JUN 84	426A

#### Page A-34:

In the listing of the "TYPE" program, change the value 4 to 5 in line 207 as follows:

```
2115:C0 05          207          CPY    #5
```



## Quality Software Products For the Apple

### BOOKS

#### **Beneath Apple ProDOS** by Don Worth & Pieter Lechner

Describes the ProDOS Operating System clearly and in detail, going beyond Apple's manuals. Many programming examples are included. 288 pages. **\$19.95**

Supplements to Beneath Apple ProDOS:

Versions 1.0.1 and 1.0.2 (combined) **\$10.00**

Version 1.1.1 **\$12.50**

Versions 1.2 and 1.3 (combined) **\$12.50**

#### **Beneath Apple DOS** by Don Worth & Pieter Lechner

The popular best seller that covers all facets of DOS 3.3 and previous Apple disk operating systems. 176 pages. **\$19.95**

#### **Understanding the Apple II** by Jim Sather

Foreword by Steve Wozniak. A definitive source of information, covers Apple II and Apple II Plus hardware, including the disk controller and logic state sequencer. 352 pages. **\$22.95**

#### **Understanding the Apple IIe** by Jim Sather

The companion to **Understanding the Apple II**, this book covers Apple IIe hardware, including video graphics and the 1985 firmware upgrade (65C02). 368 pages. **\$24.95**

### UTILITIES

#### **Bag of Tricks 2** by Don Worth & Pieter Lechner

Quality Software's popular set of Apple II disk utility programs, **Bag of Tricks**, has been thoroughly revised and updated for the ProDOS operating system. TRAX, INIT, ZAP, and FIXCAT are the four comprehensive utility programs, all with improved user interfaces to make them easier to use than the original **Bag of Tricks**.\* Unprotected diskette and 200-page manual. 64K. **\$49.95**

\*Special offer to **Bag of Tricks** owners--save \$20 by ordering directly from Quality Software. To order, send in your **Bag of Tricks** diskette and \$29.95, plus shipping, handling, and sales tax. We will return your diskette along with the new product.

#### **Universal File Conversion** by Gary Charpentier

Moves programs and data among the five operating systems used on the Apple II family of computers: DOS, ProDOS, CP/M, Pascal, and SOS. Unprotected 5 1/4" diskette and 48-page manual. 64K. **\$34.95**

**Ordering directly from Quality Software**

To order our products directly, mail this order form to Quality Software (at the address below) with your payment--the price of the software (plus sales tax if shipped to California) plus shipping and handling charges. Your payment can be a check or bank draft made payable to Quality Software in US dollars, or your VISA or MASTERCARD number and expiration date (VISA and MASTERCARD holders may phone in their orders). California residents must add the appropriate sales tax (6%, 6.5%, or 7%).

**Shipping charges:**

48 Continental United States (UPS).....\$2.50  
Alaska, Hawaii, Canada, and Mexico (air mail).....\$5.00  
All other countries (insured air mail).....\$10.00

Send your order to:

**QUALITY SOFTWARE**  
21610 Lassen Street #7  
Chatsworth CA 91311  
(818) 709-1721

QUANTITY      DESCRIPTION      AMOUNT

-----  
-----  
-----  
-----

SUBTOTAL      -----

(CA RESIDENTS) SALES TAX      -----

SHIPPING      -----

TOTAL      -----

Check # \_\_\_\_\_

OR VISA/MasterCard # \_\_\_\_\_ EXPIRES \_\_\_\_\_

Name \_\_\_\_\_

Street Address \_\_\_\_\_

City, State, Postal Code \_\_\_\_\_

Country \_\_\_\_\_

SUPPLEMENT TO

# Beneath Apple ProDOS

For ProDOS 8, Versions 1.2 and 1.3



by Don Worth and Pieter Lechner

**QS** QUALITY  
SOFTWARE