

SUPPLEMENT TO

# **Beneath Apple ProDOS**

For ProDOS Version 1.1.1

by Don D. Worth and Pieter M. Lechner

---



**QUALITY SOFTWARE**

21610 Lassen Street #7  
Chatsworth, California 91311

## Apple Books from Quality Software

<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>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.

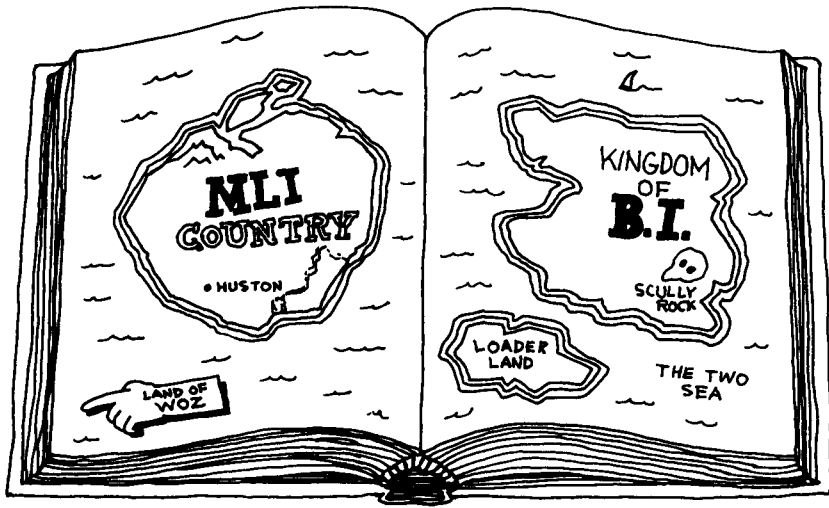
Illustrations by George Garcia

**(c)1986 Quality Software.** All rights reserved. No part of this book may be reproduced, in any way or by any means, without permission in writing from the Publisher. No liability is assumed with respect to the use of the information contained herein. While every precaution has been taken in the preparation of this book, the publisher assumes no responsibility for errors or omissions. Neither is any liability assumed for damages resulting from the use of the information contained herein.

"Apple" is a registered trademark of Apple Computer, Inc. This manual was not prepared nor reviewed by Apple Computer, Inc., and use of the term "Apple" should not be construed to represent any endorsement, official or otherwise, by Apple Computer, Inc.

## CONTENTS

<u>TOPIC</u>	<u>PAGE</u>
Introduction	5
Understanding the Listings	5
<b>PRODOS, VERSION 1.1.1</b>	
How ProDOS 1.1.1 is Loaded and Relocated	6
ProDOS Loader	7
ProDOS Relocator	10
Relocation routines	
RAMdrive Device Driver	
SYSTEM File loader	
ProDOS MLI (Kernel)	23
ProDOS System Global Page	57
ProDOS Quit Code	59
ProDOS Disk II Device Driver	63
ProDOS IRQ Handler	70
<b>BASIC.SYSTEM, VERSION 1.1</b>	
How BASIC.SYSTEM is Loaded and Relocated	71
BI Relocator	72
BASIC Interpreter (BI)	75
BI Global Page	110
<b>DISK II CONTROLLER BOOT ROM</b>	
Disk II Controller Card--Apple II/II+/IIe	112
Disk II Controller--Apple IIc	114
<b>ERRATA</b>	
Errata to <b>Beneath Apple ProDOS</b>	
1st printing, 1984	117
2nd printing, 1985	122



## A ProDOS ATLAS

## INTRODUCTION

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

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

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

## UNDERSTANDING THE LISTINGS

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

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

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

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

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

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

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

```

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

```

I-----I$FFFF
I  IRQ HANDLER I
I-----I$FF80
I-----I$FF00
I      I
I      MLI      I
I      I
I      KERNEL  I
I      I
I      (run location) I
I      I
I-----I$DE00
I      I
I-----I$D700
I  DISK II DRIVER I
I-----I$D000
I      I
I-----I$C000
I SYSTEM GLOBAL PAGE I
I-----I$BF00
I      I

I      I
I-----I$5A00
I  QUIT CODE      I
I-----I$5700
I  DISK II DRIVER I
I-----I$5000
I  IRQ HANDLER   I
I-----I$4F80
I  CLOCK CODE    I
I-----I$4F00
I SYSTEM GLOBAL PAGE I
I-----I$4E00
I      I
I      MLI      I
I      I
I      (load location) I
I      I
I-----I$2D00
I      I
I      RELOCATOR I
I-----I$2000
I      I
    
```

- 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	4F00	D742	7D
QUIT code	5700	D100*	300
RAM drive...			
Caller	2C00	FF00	9A
Driver	2A00	200**	200

```

I      I
I-----I$96C
I SYSTEM file Loader I
I-----I$800
I      I
I-----I$400
I  PAGE 3 IMAGE      I
I-----I$3D6
I      I
I-----I$C8
I 80-COL CARD CHECKER I
I-----I$80
I-----I$0
    
```

ProDOS Loader -- V1.1.1 -- 18 SEP 84      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 1.1.1 -- 18 SEP 84
*   (THE LOADER IS STILL THE SAME AS IT
*   WAS IN VERSION 1.0.1)
*
*****
*** EXTERNAL ADDRESSES ***
  
```

```

0827  ROM BOOT SUBRTN BUFFER PAGE ADDR
0828  ROM BOOT SUBRTN SLOT * 16
0829  ROM BOOT SUBRTN SECTOR TO READ
0830  ROM BOOT SUBRTN CURRENT TRACK
0831  ROM BOOT SUBRTN TRACK TO READ
0832  -- BLOCK READ PARAMETER LIST --
0833  COMMAND NUMBER (1 = READ)
0834  SLOT * 16
0835  I/O BUFFER ADDRESS ($44/$45)
0836  BLOCK TO READ ($46/$47)
0837  -----
0838  POINTER TO BLOCK READ ROUTINE
0839  VOL DIR ENTRY POINTER/FIRST INDEX PAGE
0840  ADDR OF SECOND PAGE OF INDEX BLOCK
0841  INDEX INTO INDEX BLOCK PAGES
0842  TRACK SEEK PHASE-ON INDEX
0843  TRACK PHASE WANTED
0844  BLOCK READER RETRY COUNT
0845  CURRENT TRACK PHASE/PHASE-OFF INDEX
0846  BUFFER POINTER
0847  SCREEN CENTER LINE
0848  LOAD POINT FOR RELOCATOR
0849  DISK ARM PHASE0
0850  TURN DISK DRIVE OFF
0851  TURN DISK DRIVE ON
0852  SHIFT DATA REGISTER
  
```

ProDOS Loader -- V1.1.1 -- 18 SEP 84      NEXT OBJECT ADDR: 0800  
 ADDR    DESCRIPTION/CONTENTS

```

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 $38,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.
  
```

```

0801  ***** MAIN ENTRY *****
      ON ENTRY, X = SLOT*16
      A = SECTOR NUMBER

0802  ENTRY POINT FOR APPLE II
0803  ALWAYS TAKEN (APPLE II) >>0807
0804  JUMP TO APPLE /// LOGIC >>A132
0805  SAVE SLOT*16
0806  READING SECTOR 3 NEXT?
0807  REMEMBER THIS...
0808  MAKE $CX FROM SLOT*16
0809  AND SAVE AT $49
0810  $4B/49 --> $CFFF IN ROM BOOT
0811  CHECK $CFFF
0812  BOOT ROM FOR DISK II?
0813  NO, NOT A 5.25" FLOPPY >>085B
0814  GOT BOTH SECTORS OF LOADER? >>0831
0815  NO. STOP AT SECTOR 3
0816  STORE ON PARM (0800)
0817  SKIP SECTOR 1 (GET SEC 2)
0818  DUMMY UP $CX5C AS RETURN ADDRESS
0819  AND CALL ROM SECTOR READ SUBRTN

***** LOAD PRODOS *****
      (ENTIRE LOADER IN MEMORY NOW)

0831  CURRENT TRACK IS ZERO
0832  $48/49 --> $CX00
0833  COPY A PORTION OF DISKETTE BOOT ROM
0834  TO MY BLOCK READER SUBROUTINE (0994)
0835  FROM $9F7 TO $A7E
0836  MODIFY SOME BRANCHES IN THE COPIED CODE (091D)
0837  TO SUIT MY ERROR HANDLING TASTES (0924)
0838  AND COPY SECTOR READ SUBROUTINE EXIT CODE (092B)
  
```

ProDOS Loader -- V1.1.1 -- 18 SEP 84      NEXT OBJECT ADDR: 084F  
 ADDR    DESCRIPTION/CONTENTS

```

084F TO $A7F TO $A85 (0A7F)
0855 $4B/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?
08AB 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
08EA BUMP TO NEXT BLOCK BUFFER
  
```

ProDOS Loader -- V1.1.1 -- 18 SEP 84      NEXT OBJECT ADDR: 08FE  
 ADDR    DESCRIPTION/CONTENTS

```

08EE $4E = OFFSET INTO INDEX BLOCK
08F0 GET NEXT BLOCK NUMBER FROM INDEX BLOCK
08F8 BLOCK NUMBER = 0? (END OF FILE)
08FA NOT YET, READ A BLOCK >>08E3
08FC ELSE, JUMP TO RELOCATOR AT $2000 >>2000
08FF ERROR JUMP >>093F
0902 ***** KERNEL NAME *****
0902 LENGTH OF KERNEL'S NAME
0903 'PRODOS', "PRODOS" (KERNEL NAME)
0912 ***** COPY BLOCK READ BUFFER PTR *****
0912 COPY $60/61 --> $44/45
0914 (BLOCK READ BUFFER POINTER)
091A THEN GO TO BLOCK I/O ROUTINE >>004B
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
092D AND EXIT NORMALLY
092E 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
093F ***** ERROR HANDLER *****
  
```



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

```

093F HOME CURSOR/CLEAR SCREEN <FC58>
0944 COPY "UNABLE TO LOAD PRODOS" MESSAGE (0950)
0947 TO SCREEN (05AE)
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

0986 ***** DISKETTE BLOCK READ ROUTINE *****
      $44/$45 --> BUFFER
      $46/$47 = BLOCK NO.

0986 GET BLOCK NO. LSB
0988 ISOLATE SECTOR REMAINDER
098C SKEW SECTOR BY 2
0992 AND STORE SECTOR WANTED
0994 GET MSB
0996 AND HIGH BIT OF TRACK
0999 MERGE WITH LOW PART OF TRACK
099C STORE TRACK WANTED
099F TRACK*2 IS PHASE WANTED
09A3 SET PAGE ADDRESS OF BUFFER
09A7 TURN DRIVE MOTOR ON (C089)
09AA READ SECTOR <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
09DD NO, ADJUST PHASE UP...
  
```

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

```

09D4 OR DOWN AND...
09D6 ---
09D7 SEEK ARM ONE PHASE... <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)

0A7F EXIT CODE FOR READ ROUTINES
      (COPIED HERE FROM $92B-$930)

0A86 ***** $A86-$BFF NOT USED *****
0A86 NOT USED

0C00 ***** VOLUME DIRECTORY BUFFER *****
0C00 START OF VOLUME DIRECTORY BUFFER
0C23 OFFSET TO ENTRY LENGTH FIELD
  
```

ProDOS Relocator -- V1.1.1 -- 18 SEP 84      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 1.1.1 -- 18 SEP 84  
 \*    \*    \*  
 \*\*\*\*\*

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

\*\*\*\*\* EXTERNAL ADDRESSES \*\*\*\*\*  
 0080    MACHID BUILD SUBRTN FOR 128K  
 0280    GENERAL PURPOSE BUFFER  
 0281    BUFFER+1

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

\*\*\*\*\* SCREEN LINE ADDRESSES \*\*\*\*\*  
 04B8    SCREEN BUFFER LINE  
 05A9    SCREEN BUFFER LINE  
 05B1    SCREEN BUFFER LINE  
 06A8    SCREEN BUFFER LINE  
 07A3    SCREEN BUFFER LINE  
 07AD    SCREEN BUFFER LINE  
 07D0    SCREEN BUFFER LINE

\*\*\*\*\* INTERP LOADER ADDRESSES \*\*\*\*\*  
 0800    ENTRY OF INTERP LOADER  
 08E2    'UNABLE TO FIND SYSTEM FILE'  
 090A    'INTERP FILE TOO LARGE.'  
 092A    'UNABLE TO LOAD ...'  
 093B    INTERP FILE NAME ITSELF  
 093C    +1  
 094F    LENGTH OF MESSAGE  
 0950    MLI: OPEN LIST  
 0956    MLI: GET EOF  
 0958    EOF MARK  
 0959    EOF MARK+1  
 095A    EOF MARK+2 (MSB)  
 095B    MLI: READ LIST  
 095F    READ BUFFER ADDR  
 0960    +1  
 0963    MLI: CLOSE LIST  
 0965    ' .SYSTEM'

\*\*\*\*\* MISCELLANEOUS ADDRESSES \*\*\*\*\*  
 0C00    VOLUME DIRECTORY BUFFER  
 0C23    ENTRY LENGTH  
 -- RAMDRIVE VOLUME DIRECTORY --  
 0E04    VOLUME HDR, VOLUME NAME  
 0E22    VOLUME HDR, ACCESS-TOTAL BLOCKS  
 2A00    RAMDRIVE DEVICE DRIVER LOAD ADDRESS  
 2800    DIFFERENCE OF RAMDRIVE LOAD AND RUN LOCATIONS

\*\*\*\*\* 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

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

ADDR DESCRIPTION/CONTENTS

BF99 SLOT WHICH CONTAIN CARDS WITH ROM  
 BFFF TOP OF 48K RAM

C000 \*\*\*\*\* I/O PORT ADDRESSES \*\*\*\*\*

C001 80 STORE OFF  
 C002 80 STORE ON  
 C003 READ MAIN RAM  
 C004 READ AUX RAM  
 C005 WRITE MAIN RAM  
 C006 WRITE AUX RAM  
 C007 MAIN STACK/ZERO PAGE  
 C008 ALTERNATE STACK/ZERO PAGE  
 C009 INTERNAL SLOT 3 ROM  
 C00A PERIPHERAL SLOT 3 ROM  
 C00B 80 COLUMN DISPLAY OFF  
 C00C 80 COLUMN DISPLAY ON  
 C00D READ 80STORE SWITCH  
 C00E SPEAKER  
 C00F USE MAIN MEMORY PART OF 80-COL CARD  
 C010 USE AUX MEMORY PART OF 80-COL CARD  
 C011 WRITE-ENABLE HIGH RAM  
 C012 MOTHERBOARD ROM READ ENABLE  
 C013 READ/WRITE RAM 2ND 4K BANK  
 C014 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  
 C308 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)  
 DF08 ENHANCED ROM FLAG  
 FF00 RAMDRIVE CALLER ADDRESS

\*\*\*\*\* MONITOR ROM \*\*\*\*\*

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

ADDR DESCRIPTION/CONTENTS

FB1E PADDLE READ SUBROUTINE  
 FB2F MONITOR INIT ROUTINE  
 FB33 ROM VERSION BYTE  
 FBC0 SECONDARY VERSION BYTE (0-3)  
 FC58 CLEAR SCREEN  
 FE84 SET NORMAL VIDEO  
 FE89 IN#0  
 FE93 PR#0

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

2000 STORE SLOT IN MLI ONLINE PARMS  
 2005 PRINT "APPLE II PRODOS..." <2499>  
 2008 SET UP FOR COMMON MOVES (220A)  
 200E RELOCATE SOME ROUTINES & DATA TO LOW MEMORY <26A6>  
 2011 ERROR? >>2036  
 2013 NO, PROCEED...  
 2017 BE SURE 48K OF MAIN MEMORY EXISTS (BFFF)  
 201E IF NOT, ERROR >>208E  
 2026 MAKE DOUBLY SURE >>208E  
 2028 SELECT MOTHERBOARD ROMS (C082)  
 202B DETERMINE MACHINE TYPE <2402>  
 2030 PICK UP CONFIGURATION BYTE  
 2032 64K OR MORE MEMORY?  
 2034 YES, WE HAVE 64K RAM >>2039  
 2036 ERROR. MUST HAVE 64K FOR PRODOS 1.1.1!! >>21C3

\*\*\*\*\* RELOCATE PRODOS \*\*\*\*\*

2039 SET UP FOR MLI MOVE (220C)  
 203F COPY/RELOCATE PRODOS ITSELF <26A6>  
 2042 ERROR? >>208E  
 2044 ENABLE MOTHERBOARD ROMS AGAIN (C082)  
 2047 CHECK ROM I.D. BYTE (FBB3)  
 204A APPLE //e FAMILY?  
 204C NO, LEAVE I.D. BYTE AS IS >>206D  
 2050 TEST ANOTHER ROM I.D. BYTE (FBC0)  
 2053 SAVE BIT TEST RESULTS  
 2054 GET MACHID  
 2056 STRIP BITS THAT IDENTIFY MODEL  
 205B IT'S A //e IF BITS 6 & 7 ARE HIGH >>2069

---  
 205D EITHER A //c OR A FUTURE SYSTEM  
 205E CHECK HIGH BITS OF \$FBC0 AGAIN  
 2060 CHECK HIGH BITS OF \$FBC0 AGAIN  
 2061 BIT 7 ON? >>2067  
 2063 YES, FUTURE SYSTEM.  
 2067 IF BIT 6 ON, IT'S A FUTURE SYSTEM. >>206B

---  
 2069 REPLACE UPDATED MACHID  
 206B COPY BOOT DEVICE ID TO READ BLOCK PARMS (21FE)  
 206D

ProDOS Relocator -- V1.1.1 -- 18 SEP 84      NEXT OBJECT ADDR: 2073  
 ADDR      DESCRIPTION/CONTENTS

2073 AND AS LAST DEVICE USED (BF30)  
 2076 DETERMINE PERIPHERAL CARD CONFIGURATION <252A>  
 2079 BOOT DEVICE TO... (2205)  
 207C GLOBAL PAGE LAST DEVICE USED (BF30)  
 2082 WRITE ENABLE BANK1 OF HIGH RAM (C08B)  
 208B COPY CLOCK CODE TO DEVICE DRIVER AREA <26A6>  
 208E ERROR? >>20BF  
 2090 CHECK MACHINE TYPE AGAIN (BF98)  
 2093 GOT 64K OR MORE?  
 2097 NO >>20C2  
 2099 YES, QUIT VECTOR ---> \$FCE5  
 20A3 WRITE TO HIGH RAM (BANK2) (C083)  
 20AC POINT TO QUIT CODE TABLE (2211)  
 20AF MOVE QUIT CODE TO HIGH RAM <26A6>  
 20B4 STORE QUIT VECTOR START PAGE (D000)  
 20B7 WRITE TO HIGH RAM (BANK1) (C08B)  
 20BA AGAIN (C08B)  
 20BF RELOCATION ERROR >>21C3  
 20C2 GET MACHID YET AGAIN (BF98)  
 20C5 128K?  
 20C9 NO... >>20D1  
 20CE YES, ESTABLISH RAM DRIVE IN UPPER 64K <28FF>

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

20D1 READ ROM (C081)  
 20D4 GET IRQ VECTOR FROM ROM (FFFF)  
 20DA CARRY CLEAR IF IRQ VECTOR IN C3 ROM  
 20DE IT'S AN OLD ROM >>20FD  
 20E0 READ & WRITE RAM (BANK1) (C08B)  
 20E6 SWITCH TO AUX HIGH RAM (C009)  
 20E9 PUT IRQ VECTOR IN AUX HIGH RAM (FFFF)  
 20EF BACK TO MAIN HIGH RAM, Z-PAGE (C008)  
 20F2 PUT IRQ VECTOR IN MAIN HIGH RAM (FFFF)  
 20F8 SET FLAG INDICATING  
 20FA ENHANCED IRQ LOGIC ON BOARD (DFD8)

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

20FD ENABLE INTERNAL VIDEO FIRMWARE (C00A)  
 2100 CHECK FOR ROM (BF99)  
 2103 IN SLOT 3.  
 2105 NONE THERE >>216D  
 2107 LOOK AT THE SLOT 3 ROM (C00B)  
 210A AT OFFSET +\$05 (C305)  
 210D THERE MUST BE A \$38  
 2111 AND AT OFFSET +\$07 (C307)  
 2114 THERE MUST BE AN \$18  
 2118 AND AT OFFSET +\$0B (C30B)  
 211B THERE MUST BE A 1  
 211F AND AT OFFSET +\$0C (C30C)

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

2124 INDICATE AN 80-COL CARD.  
 2128 CHECK MACHINE TYPE (BF98)  
 212D IS THIS AN APPLE III?  
 212F OK, IT'S GOT 80-COL CAPABILITY >>2165  
 2131 OTHER MANUFACTURERS MUST FOLLOW THE RULES! (C3FA)  
 2134 MUST HAVE BIT INSTRUCTION AT \$C3FA  
 2136 GOOD BOY, YOU FOLLOWED THE RULES! >>2165  
 2138 GIVE CONTROL BACK TO MOTHERBOARD ROM (C00A)  
 213B TURN ON 80-COL (C001)  
 213E CHECK FOR AUX MEM. (C055)  
 2143 PUT A BYTE AT AUX \$400 (0400)  
 2146 THE ACCUMULATOR LEFT  
 2147 AND DO THE SAME WITH \$400 (0400)  
 214A STILL THE SAME? (0400)  
 214D NO, NO 80-COL MEMORY >>2156  
 214F SHIFT TO THE RIGHT  
 2153 STILL THE SAME? (0400)  
 2156 BACK TO MAIN MEMORY (C054)  
 2159 TURN OFF 80-COL (C000)  
 215E WAS 80-COL MEMORY FOUND? >>2165  
 215C NO, SO TURN OFF 80-COL FLAG (BF98)  
 2161 IN MACHINE I.D. BYTE.  
 2163 ALWAYS BRANCH >>216A  
 2165 TURN ON 80-COL FLAG (BF98)

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

216D MLI: ONLINE DEVICE CALL <BF00>  
 2173 ERROR? >>21C3  
 2178 VALID VOLUME NAME?  
 217A IF NOT, ERROR >>21C3  
 217D ELSE, BUMP LENGTH BY ONE  
 2182 AND PREFIX NAME BY A "/"  
 2187 MLI: SET PREFIX <BF00>  
 218D ERROR? >>21C3

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

218F ---  
 2191 \$14/15 --> \$C00  
 2197 ---  
 219C BLOCK = 2 (VOLUME DIRECTORY) (2208)  
 21A2 MLI: READ BLOCK <BF00>  
 21AB ERROR? >>21C3  
 21AC GET NEXT BLOCK NUMBER  
 21B2 IF ZERO, END OF VOLUME DIRECTORY >>21C0  
 21BA ADD TWO PAGES (ONE BLOCK) TO POINTER  
 21BC AND STOP AT \$1400 IN ANY CASE  
 21BE ELSE, READ NEXT BLOCK AS WELL >>2197  
 21C0 WHEN DONE, JUMP TO SYSTEM FILE LOADER >>0800

ProDOS Relocator -- V1.1.1 -- 18 SEP 84 NEXT OBJECT ADDR: 21C0

ADDR DESCRIPTION/CONTENTS

21C3 \*\*\*\*\* ERROR HANDLER \*\*\*\*\*  
 21C6 ENABLE MOTHERBOARD ROMS (C082)  
 21C6 CLEAR SCREEN <FC58>  
 21CB PRINT "RELOCATION/CONFIG ERROR" (21D7)  
 21D4 THEN SLEEP FOREVER >>21D4

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

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

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

2201 MLI: SET PREFIX PARMS  
 2202 PREFIX IS AT \$280

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

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

2212 \*\*\*\*\* RELOCATION TABLES \*\*\*\*\*  
 +0: 00 - ZERO BLOCK OF MEMORY  
 01 - COPY BLOCK  
 02 - RELOCATE MSB ADDRESSES  
 03 - RELOCATE 2 BYTE ADDRS  
 04 - RELOCATE INSTRUCTIONS  
 +1/2: ADDR OF OUTPUT BLOCK  
 +3/4: LENGTH OF BLOCK IN BYTES  
 +5/6: ADDR OF INPUT BLOCK (IF ANY)  
 +7: NUM RANGES TO CORRECT FOR (-1)  
 +8: START PAGES  
 +8+COUNT: END PAGE ADDRESSES  
 +8+COUNT+COUNT+ADDITIVE CORRECTION FACTOR  
 \*\*\*\*\* COMMON MOVES TABLE \*\*\*\*\*

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

ProDOS Relocator -- V1.1.1 -- 18 SEP 84 NEXT OBJECT ADDR: 221A

ADDR DESCRIPTION/CONTENTS

221A TO =\$3D6  
 221C LEN=\$2A  
 221E FRM=\$23D8  
 2220 COPY (CHECKSUM)  
 2221 TO =\$0A  
 2223 LEN=\$02  
 2225 FRM=\$14  
 2227 COPY (CHECK FOR 80-COL CARD)  
 2228 TO =\$80  
 222A LEN=\$48  
 222C FRM=\$2451  
 222E END OF TABLE

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

222F COPY (QUIT CODE)  
 2230 TO =\$D100  
 2232 LEN=\$300  
 2234 FRM=\$5700  
 2236 END OF TABLE

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

2237 COPY (IRQ HANDLER)  
 2238 TO =\$FF80  
 223A LEN=\$80  
 223C FRM=\$4F80  
 223E COPY (SYSTEM GLOBAL PAGE)  
 223F TO =\$BF00  
 2241 LEN=\$100  
 2243 FRM=\$4E00  
 2245 ZERO (PRODOS KERNEL DATA AREA)  
 2246 ADR=\$D700  
 2248 LEN=\$700  
 224A COPY (PRODOS KERNEL)  
 224B TO =\$DE00  
 224D LEN=\$2100  
 224F FRM=\$2D00  
 2251 COPY (DISKETTE DRIVER)  
 2252 TO =\$D000  
 2254 LEN=\$700  
 2256 FRM=\$5000  
 2258 END OF TABLE

\*\*\*\*\* PRODOS CLOCK TABLE \*\*\*\*\*

2259 COPY (CLOCK CODE)  
 225A TO =\$D742  
 225C LEN=\$7D  
 225E FRM=\$4F00  
 2260 RELOCATE INSTRUCTIONS

ProDOS Relocator -- V1.1.1 -- 18 SEP 84      NEXT OBJECT ADDR: 2261  
 ADDR    DESCRIPTION/CONTENTS

2261            TO =\$D742  
 2263            LEN=\$69  
 2265            FRM=\$D742  
 2267            FOR ADORS=\$CLXX-\$CLXX  
 226A            ADJUST BY=\$50  
 226B            END OF TABLE

226C \*\*\*\*\* SYSTEM FILE LOADER \*\*\*\*\*  
 (COPIED TO AND RUN AT \$800)

226C \$10/11 --> VOLUME DIRECTORY ENTRIES  
 226E INITIALLY AT \$C00  
 2270 OFFSET BEYOND LINKS (+4)  
 2272 (TURN NEXT INSTRUCTION INTO BIT)

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

2273 PICK UP LSB  
 2276 BUMP BY ENTRY LENGTH (0C23)  
 2279 UPDATE LSB  
 227B PAGE OVERFLOW? >>228F  
 227D NO, ROOM FOR ONE MORE ENTRY? (0C23)  
 2282 NO, CHECK MSB  
 2285 START OF A BLOCK? >>2291  
 2287 NO, AT END OF DIRECTORY?  
 2289 YES, FILE NOT FOUND IN DIRECTORY >>22A9  
 228B NO, START NEW BLOCK AT +4  
 228D AND UPDATE LSB  
 228F BUMP MSB

2291 ---  
 2295 CHECK FILE TYPE FOR PRODOS "SYS" FILE  
 2297 NOT IT? >>2273  
 229A INACTIVE ENTRY?  
 229C IF SO, SKIP IT >>2273  
 22A0 SAVE NAME LENGTH AT \$280 (0280)  
 22A5 MUST BE AT LEAST 8 CHARS LONG >>2273  
 22A7 JUMP AROUND ERROR CODE >>22AC  
 22A9 ERROR - SYSTEM FILE NOT FOUND >>2319  
 22AB HARD BREAK IN THAT CASE  
 22AC ---  
 22AF IS THIS ".SYSTEM"?  
 22B1 (SEE \$23D1) (0965)  
 22B5 NO, SKIP ENTRY >>2273  
 22B9 CHECK ALL CHARACTERS IN NAME >>22AF

\*\*\*\*\* LOAD SYSTEM FILE AT \$2000 \*\*\*\*\*

ProDOS Relocator -- V1.1.1 -- 18 SEP 84      NEXT OBJECT ADDR: 22BB  
 ADDR    DESCRIPTION/CONTENTS

22BB ---  
 22BD ---  
 22BE COPY NAME TO \$281  
 22C5 AND TO "UNABLE TO LOAD" MSG (093B)  
 22CD ADD BLANK AT END OF NAME  
 22CF IN MESSAGE (093C)  
 22D3 NAMELEN + ERRORSNGLEN  
 22D5 SAVE AT \$23BB (094F)  
 22D8 MLI: OPEN .SYSTEM FILE <BF00>  
 22DC (PARM LIST AT \$24BC)  
 22DE ERROR? >>2326  
 22E0 MLI: GETEOF <BF00>  
 22E4 (PARM LIST AT \$23C2)  
 22E6 ERROR? >>2326  
 22E8 GET MSB (SEE \$23C6) (095A)  
 22EB BIGGER THAN 64K??? >>2340  
 22F0 MUST BE LESS THAN \$9800 BYTES  
 22F2 OR ERROR... >>2340

22F4 STORE LENGTH IN MLI READ LIST (0960)  
 22FA AND LSH TOO (095F)  
 22FD MLI: READ SYSTEM FILE INTO \$2000 <BF00>  
 2301 (PARM LIST AT \$23C7)  
 2303 NO ERRORS? >>230B  
 2305 ERROR, BAD BUFFER?  
 2307 YES, FILE WAS TOO LARGE >>2340  
 2309 ELSE, "UNABLE TO LOAD .." >>2326

230B MLI: CLOSE SYSTEM FILE <BF00>  
 230F (PARM LIST AT \$23CF)  
 2311 ERROR? >>2326  
 2313 NO, ENABLE MOTHERBOARD ROMS (C082)  
 2316 AND JUMP TO BEGINNING OF FILE >>2000

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

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

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

ProDOS Relocator -- V1.1.1 -- 18 SEP 84      NEXT OBJECT ADDR: 233E  
 -----  
 ADDR    DESCRIPTION/CONTENTS  
 -----

ProDOS Relocator -- V1.1.1 -- 18 SEP 84      NEXT OBJECT ADDR: 23FA  
 -----  
 ADDR    DESCRIPTION/CONTENTS  
 -----

```

2340 ---
2342 PRINT "SYSTEM PROGRAM TOO LARGE" (090A)
234B GO TO SLEEP FOREVER >>234E

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

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

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

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

23CF MLI: CLOSE LIST
23D0 REFNUM=0, CLOSE ALL FILES

23D1 '.SYSTEM'

23D8 ***** END OF SYSTEM FILE LOADER *****
23D8 ***** PAGE 3 VECTOR IMAGE *****
      ($3D6-$3FF)
      (INCLUDES A ROUTINE AT $3D6 THAT COPIES
      CRITICAL ZERO PAGE VALUES TO AUX MEM)

23D8 FROM MAIN Z-PAGE, (C008)
23DB GET X+1 VALUES STARTING AT $42
23DD AND PUT IN AUX Z-PAGE (C009)
23E0 AT SAME LOCATION.
23E5 "NO DEVICE CONNECTED" ERROR
23E8 BACK TO MAIN Z-PAGE (C008)
23EB RETURN
23EC ADDRESS OF MLI ROUTINE
23F2 BRK HANDLER AT $FA59
23F4 RESET AT $FF59
23F6 POWER UP BYTE
23F7 & VECTOR TO $FF59 >>FF59
  
```

```

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

2402 ***** DETERMINE MACHINE ID *****
      $0C=00.. 0... APPLE II
      01.. 0... APPLE II+
      10... 0... APPLE IIE
      10.. 1... APPLE IIC
      11.. 0... APPLE /// EMULAT.
      ..01 ..... 48K RAM
      ..10 ..... 64K RAM
      ..11 ..... 128K RAM
      .....1.. 80 COL CARD
      .....1.. THUNDER CLOCK

2402 ASSUME NOTHING AT FIRST
2406 GET A ROM BYTE (FBB3)
2409 APPLE II?
240B YES, SET BIT >>242E
240D NO,
240F APPLE IIE?
2411 YES, SET BIT >>242E
2413 NO,
2415 APPLE II+?
2417 NO, WHAT IS IT? >>2428
241C REALLY A II+?
241E YES >>242E
2422 /// EMULATION MODE?
2426 ---
2427 RETURN
2428 OTHERWISE, UNKNOWN MACHINE
242A CREATE INVALID INSTR AT $80
242C AND GO THERE >>244E
242E UPDATE MACHID
2433 READ/WRITE ENABLE HIGH RAM (BANK1) (C08B)
2438 SEE IF HIGH RAM EXISTS (D000)
244A IF PRESENT, MARK IN MACHID

2451 ***** LOOK FOR 64K OF AUX RAM *****
      (CODE MOVED TO $80 TO ALLOW BANK SWITCH)
      (ENTERED WITH MACHID IN ACCUMULATOR)

2451 UPDATE MACHID
2453 IIE? >>248A
2455 YES,
2457 BANK TO AUX MEMORY (C005)
245D STORE A PATTERN AT $C00 (0C00)
2460 AND AT $800 (0800)
2466 MAKE SURE PATTERN STAYS THERE
2468 IT DIDN'T! >>2478
  
```

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

246A NOW SHIFT \$C00 TO THE LEFT (C0C00)  
 246D AND SHIFT THE ACCUM TO THE LEFT  
 246E ARE THEY STILL THE SAME? (C0C00)  
 2471 NO, AUX RAM NOT THERE. >>2478  
 2473 DID \$800 MOVE TOO? (08000)  
 2476 NO, SO WE HAVE FULL 128K! >>247B  
 2478 DON'T HAVE 128K  
 247B ---  
 247C BANK BACK TO MAIN MEMORY (C004)  
 2482 64K? >>248A  
 2486 NO, INDICATE 128K  
 2488 IN MACHID  
 248A SET UP \$A/B --> "APPLE II"  
 248D IN MOTHERBOARD ROM  
 248F AT \$FB09  
 2491 BUT DO IT IN A CONVOLUTED WAY  
 2498 RETURN TO CALLER

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

2499 CLICK SPEAKER (C030)  
 249C STORE IN MAIN MEMORY (C00C)  
 249F 80 COL DISPLAY OFF (C000)  
 24A2 SET NORMAL VIDEO <FE84>  
 24A5 CALL MONITOR INITIALIZATION <FB2F>  
 24A8 SET VIDEO PR#0 <FE93>  
 24AB SET KEYBD IN#0 <FE89>  
 24AE OUT OF DECIMAL MODE  
 24AF DISABLE FOR INTERRUPTS  
 24B0 CLEAR SCREEN <FC58>  
 24B5 PRINT "APPLE //" (24E3)  
 24C0 PRINT "PRODOS 1.1.1 ETC." (24EB)  
 24CB PRINT A BLANK AT \$6A8 (2502)  
 24D6 PRINT "COPYRIGHT ETC." (2503)  
 24DF CLICK SPEAKER AGAIN (C030)  
 24E2 DONE

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

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

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

252A ---  
 252C ZERO SOME THINGS  
 2533 NO DISKS ACTIVE YET (BF31)  
 2538 \$10/11 --> \$C700 (LOOP THRU ALL SLOTS)  
 253A RESET I/O CARD ROMS (CFFF)

ProDOS Relocator -- V1.1.1 -- 18 SEP 84      NEXT OBJECT ADDR: 253F  
 ADDR    DESCRIPTION/CONTENTS

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

\*\*\*\*\* PROFILE FOUND \*\*\*\*\*

2553 ELSE, SAVE AS LSB OF BLOCK READ SUBRTN  
 2555 GET \$CSFE (STATUS BYTE)  
 2558 CAN WE AT LEAST READ STATUS AND DATA?  
 255C YES? >>2563  
 255E NO.  
 2561 NOT A DISK AFTER ALL >>25AD  
 2563 GET STATUS BYTE AGAIN  
 2567 TOP NIBBLE IS DEVICE ID  
 2568 PROFILE SHOULD BE \$04  
 256A CHECK NUMBER OF VOLS (SHOULD BE 0)  
 256B GET SLOT NO. FOR DEVICE DRIVER LOC.  
 256D AND GO DO COMMON PROCESSING FOR DISK >>2579

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

256F \$12 ZERO FOR DISK II  
 2571 GET DISK II DEVICE DRIVER LOCATION (266A)  
 2575 (\$F800 OR \$B800) (266B)  
 2578 DISK II HAS 2 DRIVES

\*\*\*\*\* DISK FOUND \*\*\*\*\*

2579 SAVE DEVICE ADDRESS  
 257B SET UP INDEX OF SLOT\*2  
 2583 BUILD ST (S=SLOT, T=0 DISKII,4 PROFILE)  
 2586 BUMP DEVICE COUNT BY ONE (BF31)  
 258A AND ADD DRIVE TO SYSTEM SEARCH LIST (BF32)  
 258E NUMBER OF DRIVES  
 2590 ONLY ONE? >>2596  
 2592 NO, BUMP INDEX  
 2593 AND MARK SECOND DRIVE IN SEARCH LIST (BF32)  
 2596 STORE FINAL DEVICE COUNT (BF31)  
 259B SET UP DISK DEVICE DRIVER VECTORS (BF11)  
 259E IN SYSTEM GLOBAL PAGE >>25A8  
 25A0 (SET UP TWO VECTORS FOR A DISK II) (BF21)  
 25A8 ---  
 25AC I RECOGNIZE THIS CARD  
 25AD GO MARK SLTBYT TO SHOW ROMS IN SLOT <25FA>  
 25B4 DO ALL CARDS EXCEPT  
 25B6 SLOT 0 (\$C000) >>253A  
 25BC GET LAST DISK DEVICE IN SEARCH LIST (BF32)  
 25C2 BOOT DRIVE? (BF30)



ProDOS Relocator -- V1.1.1 -- 18 SEP 84 NEXT OBJECT ADDR: 25C6

```

ADDR  DESCRIPTION/CONTENTS
-----
25C6 NO, KEEP LOOKING >>25CA
25CA ---
25CD GET DEVICE COUNT (BF31)
25D1 IS BOOT DRIVE IN LIST? >>25E7
25D3 SO IT WILL BE SEARCHED FIRST... (BF30)
25D6 STORE BOOT AT END OF SEARCH LIST (BF32)
25DA ANY OTHERS? >>25EE
25DD YES, SECOND DRIVE? >>25E7
25E1 STORE IT RIGHT BEHIND BOOT DRIVE (BF32)
25E5 NOW ANY MORE? >>25EE
25E7 ---
25E8 YES, MOVE OTHERS AHEAD IN LIST (BF32)
25EE DO CHECKSUM ON ROM <267C>
25F1 NOT AN AUTOSTART ROM? >>25F7
25F3 AUTOSTART, STORE FINISHED MACHID (BF98)
25F6 AND LEAVE
25F7 NONAUTOSTART, UNKNOWN MACHINE, SO CRASH! >>2428

```

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

```

25FA ***** IDENTIFY I/O CARD *****
25FA DO WE ALREADY RECOGNIZE THIS CARD? >>265B
25FC NO,
25FE CHECK SIGNATURE ON CARD FOR THUNDER CLOCK
2603 NOT IT? >>261F
2609 THUNDER CLOCK, WHICH SLOT?
260B SAVE SLOT NUMBER (LESS 1)
260D IN CLOCK CODE RELOCATION TABLE (226A)
2612 ENABLE CLOCK/CALENDAR JUMP IN GLOBALS (BF06)
2617 IS THERE A MACHID? >>25EE
2619 IF SO, MARK THAT A CLOCK IS PRESENT
261B AND UPDATE MACHID
261D GO MARK ROM IN THIS SLOT >>265B
261F ---
2621 CHECK SIGNATURE OF MYSTERY CARD
2623 STANDARD BASIC SUPPORTED?
2625 NO, UNKNOWN CARD >>264A
2629 YES,
262B DOUBLE CHECK BASIC SUPPORTED
262D NO, UNKNOWN CARD >>264A
2631 YES,
2633 GENERIC SIGNATURE?
2635 NO, UNKNOWN CARD >>264A
2638 YES,
263C 80 COLUMN CARD?
263E NO, UNKNOWN CARD >>264A
2642 GET MACHID IF WE HAVE ONE >>25EE
2644 MARK 80 COLUMN CARD PRESENT
2646 AND UPDATE MACHID
2648 GO MARK ROM ON CARD PRESENT >>265B
264A UNKNOWN CARD, CHECK ROM TO...
264E SEE IF IT WILL HOLD A VALUE...

```

ProDOS Relocator -- V1.1.1 -- 18 SEP 84 NEXT OBJECT ADDR: 2654

```

ADDR  DESCRIPTION/CONTENTS
-----
2654 FOR SOME TIME.
265B IF SO, WE HAVE A CARD IN SLOT
265D CONVERT SLOT NUMBER...
2660 TO A BIT POSITION (2674)
2663 AND OR INTO SLTBYT (BF99)
2669 RETURN TO CALLER
266A ***** DATA AREA *****
266A DISK DEVICE DRIVER ENTRY POINT
266B (2 BYTE ADDRESS)
266C DEVICE SIGNATURE FOR:
266E +0,+2,+4,+6 = THUNDERCLOCK
2670 +1,+3,+5,+7 = DISK
2672 (+7 NOT CHECKED)
2674 BIT POSITION TABLE FOR SLOTS
2677 (ALSO USED IN CHECKSUM CALCS)

```

267C \*\*\*\*\* COMPUTE AUTOSTART ROM CHECKSUM \*\*\*\*\*

```

267C ---
267D GET ZERO IN INDEX REGISTER (2674)
2680 SUM $FB09 ("APPLE II") IN ROM
2687 UPDATE CHECKSUM (2674)
268E DO 8 BYTES IN ALL (2677)
2694 MOVE LENGTH TO HIGH NIBBLE
2699 AND COMBINE WITH CHECKSUM (2674)
269C FUDGE FACTOR
269E SHOULD COME OUT ZERO >>26A3
26A0 IT DID...RETURN WITH MACHID
26A2 RETURN
26A3 ELSE, RETURN WITH ZERO MACHID
26A5 RETURN
26A6 ***** RELOCATION ROUTINE *****
      (X/Y REGS CONTAIN TABLE ADDR)

```

```

26A6 SAVE PASSED TABLE ADDRESS
26AA ---
26AC GET OPERATION CODE
26AE VALID OPERATION? (4 OR LESS)
26B0 NO, ERROR >>2724
26B4 $14/15 --> OUTPUT BLOCK
26BE $16/17 --> LENGTH
26C7 NEGATIVE LENGTH? >>2726
26C9 CHECK OPERATION CODE
26CA ZERO BLOCK? >>272F
26CD NO, $12/13 = $18/19 --> INPUT BLOCK
26D7 $1A/1B --> END OF INPUT BLOCK

```

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

26E4 COPY BLOCK ONLY? >>2753  
 26E6 SAVE RELOCATION OPERATION CODE (287F)  
 26EC SAVE NUMBER OF RANGES TO CHECK (2880)  
 26F0 ---  
 26F1 COPY START PAGES TO TABLE  
 26FC ---  
 26FD AND END PAGES  
 2708 ---  
 2709 AND FINALLY, RELOCATION FACTORS  
 2711 BUMP TO NEXT TABLE ENTRY <2759>  
 2714 RESTORE OPERATION CODE (287F)  
 2719 RELOCATE INSTRUCTIONS? >>2729

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

2729 \*\*\*\*\* 4 - RELOCATE INSTRUCTIONS \*\*\*\*\*  
 2729 RELOCATE INSTRUCTIONS <27CF>  
 272C AND THEN COPY BLOCK >>271E

272F \*\*\*\*\* 0 - ZERO BLOCK \*\*\*\*\*  
 272F BUMP TABLE POINTER TO NEXT ENTRY <2759>  
 2734 GET NUMBER OF PAGES TO DO  
 2736 NO FULL PAGES? >>2744  
 2739 ZERO AN ENTIRE PAGE  
 273E BUMP PAGE POINTER  
 2740 AND DECREMENT LENGTH  
 2744 GET LENGTH OF PARTIAL LAST PAGE  
 2746 NO PARTIAL PAGE? >>2750  
 2749 ZERO PARTIAL PAGE TOO  
 2750 DONE, GET NEXT TABLE ENTRY >>26AA

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

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

ProDOS Relocator -- V1.1.1 -- 18 SEP 84      NEXT OBJEC' ADDR: 2759  
 -----  
 ADDR    DESCRIPTION/CONTENTS  
 -----

2759 ADD FINAL ENTRY INDEX..  
 275D TO TABLE ENTRY ADDRESS  
 2765 RETURN

2766 \*\*\*\*\* COPY BLOCK \*\*\*\*\*

2766 ---  
 276A INPTR < OUTPTR? >>2777  
 276C NO, GREATER? >>279A  
 276E MSB'S ARE EQUAL, CHECK LSB'S ALSO  
 2776 EXIT IF EQUAL  
 2777 INPTR < OUTPTR, COPY LAST PAGES FIRST  
 277B BUMP BOTH INPTR AND OUTPTR BY..  
 277D LENGTH-1 TO POINT AT LAST BYTE  
 2785 START WITH SHORT LAST PAGE LENGTH  
 2789 ---

278A COPY BYTES BACKWARDS THROUGH MEMORY  
 2791 DROP ADDRESSES AND LENGTH BY 256  
 2797 AND CONTINUE UNTIL FINISHED >>2789  
 2799 RETURN

279A INPTR > OUTPTR, COPY PAGES FORWARD  
 279C HOW MANY FULL PAGES LEFT?  
 279E NONE? >>27AF  
 27A0 COPY A FULL PAGE  
 27A7 AND BUMP ADDRESSES  
 27AB DECREMENT LENGTH BY 256  
 27AD AND DO ALL PAGES >>27A0  
 27AF GET LENGTH OF LAST PAGE  
 27B1 EVEN PAGE BOUNDARY? >>27BC  
 27B3 NO, COPY SHORT LAST PAGE  
 27BC RETURN

27BD \*\*\*\*\* ADDR/PAGE RELOCATE \*\*\*\*\*

27BD GET TABLE ENTRY TYPE (287F)  
 27C1 GET PAGE TO RELOCATE  
 27C3 RELOCATE A SINGLE ADDRESS <27FB>  
 27C6 BUMP BY 1 OR 2 BYTES (287F)  
 27C9 ADVANCE POINTER <2817>  
 27CC AND CONTINUE UNTIL COMPLETE >>27BD  
 27CE RETURN

27CF \*\*\*\*\* INSTRUCTIONS RELOCATE \*\*\*\*\*

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

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

27DA NO >>27E3  
 27DC YES, 3 BYTE ADDRESS TO CORRECT  
 27DE RELOCATE ADDRESS <27FB>  
 27E1 AND ADVANCE BY 3 BYTES  
 27E3 NEXT INSTRUCTION <2817>  
 27E6 CONTINUE UNTIL FINISHED >>27CF  
 27E8 RETURN  
 \*\*\*\*\* INVALID OPCODE \*\*\*\*\*  
 27E9 POP THE STACK  
 27EB RETURN WITH POINTER TO BAD INSTRUC.  
 27EF DIE HORRIBLY  
 27F2 RETURN

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

27FB \*\*\*\*\* RELOCATE ABSOLUTE ADDRESS \*\*\*\*\*  
 27FB GET PAGE NUMBER TO CHECK  
 27FD GET NUMBER OF RANGES (LESS ONE) (2880)  
 2800 IS IT PRIOR TO START OF THIS RANGE? (2881)  
 2803 YES? >>280C  
 2805 NO, IS IS AFTER END OF RANGE? (2889)  
 2808 NO? >>2810  
 280C ---  
 280D CHECK EACH RANGE >>2800  
 280F RETURN  
 ---  
 2810 ADD FUDGE FACTOR TO ADDRESS (2891)  
 2814 AND UPDATE IT  
 2816 RETURN

2817 \*\*\*\*\* BUMP POINTER TO NEXT ADDR \*\*\*\*\*  
 ---  
 2817 ADD LENGTH TO POINTER  
 281F CHECK TO SEE IF WE ARE DONE  
 2825 ---  
 2829 RETURN

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

282A \*\*\*\*\* COMPUTE INSTRUCTION LENGTH \*\*\*\*\*  
 282A A-REG CONTAINS OPCODE  
 282B ISOLATE LAST TWO BITS FOR LATER  
 2830 USE LAST 6 BITS AS TABLE INDEX  
 2832 GET BYTE WITH 4 LENGTHS IN IT (283F)  
 2835 ---  
 2836 USING TOP TWO BITS AS INDEX... >>283C  
 2838 SHIFT DOWN THE PROPER LENGTH  
 283C AND ISOLATE IT IN A-REG  
 283E RETURN

283F \*\*\*\*\* 6502 OP LENGTH TABLE \*\*\*\*\*  
 EACH BYTE CONTAINS FOUR 2 BIT LENGTHS

283F ---  
 287F \*\*\*\*\* RELOCATION DATA \*\*\*\*\*

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

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

28FF \*\*\*\*\* SET UP RAMDRIVE IN AUXMEM \*\*\*\*\*  
 (THIS ROUTINE PUTS THE RAMDRIVE DEVICE DRIVER  
 IN MEMORY, PUTS THE ADDRESS OF THE DRIVER  
 IN THE DEVICE DRIVER ADDRESS LIST, AND  
 ADDS THE RAMDRIVE TO THE ONLINE DEVICE LIST.)

28FF SUBROUTINE STARTS WITH NOP  
 2902 RELOCATE RAMDRIVE CALLER NOW AT.. (2C00)  
 2905 TO HIGH RAM AT.. (FF00)  
 290D NOW PREPARE TO MOVE  
 290F RAMDRIVE DEVICE DRIVER  
 2911 INTO AUX RAM AT \$200.  
 2914 \$3C/\$3D --> \$2A00  
 2918 \$3E/\$3F --> \$2BFF  
 291D \$42/43 --> \$200  
 2923 COPY MAIN MEM TO AUX MEM  
 2924 USE AUXMOVE TO COPY IT <C311>  
 2929 SLOT 3, DRIVE 2 DEVICE DRIVER.. (BF26)  
 292C IS AT \$FF00  
 2931 BUMP DEVICE COUNT (BF31)  
 2937 ADD DEVICE TO ONLINE DEVICE LIST  
 293C RETURN

ProDOS Relocator -- V1.1.1 -- 18 SEP 84 NEXT OBJECT ADDR: 293C  
 ADDR DESCRIPTION/CONTENTS

293D \*\*\*\*\* 293D-29FF NOT USED \*\*\*\*\*  
 293D ---  
 2989  
 2A00 \*\*\*\*\* RAMDRIVE (/RAM) DEVICE DRIVER \*\*\*\*\*  
 (COPIED TO AND RUN AT \$200 IN AUX RAM)  
 (THIS IS THE MAIN PART OF THE DEVICE DRIVER.  
 IT IS CALLED BY THE RAMDRIVE CALLER  
 WHICH IS LOCATED AT \$FF00 IN MAIN MEMORY.)  
 2A00 SAVE THE \$0STORE SETTING (C018)  
 2A04 FORCE RAM READ/WRITE (C000)  
 2A09 COPY INPUT PARAMETERS  
 2A0B TO AUX PAGE 3. (03BD)  
 2A11 FIRST TIME IN OR FORMAT COMMAND? (03BC)  
 2A14 NO, SKIP FORMAT LOGIC >>2A4F

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

\*\*\*\*\* READ/WRITE RAMDRIVE BLOCK \*\*\*\*\*  
 2A4F CONVERT BLOCK NUMBER TO PAGE NUMBER (03C1)  
 2A55 THIS PAGE IN HIGH RAM?  
 2A57 YES >>2A63  
 2A59 NO, IS IT BLOCK 3? (VOLUME BIT MAP)  
 2A5B NO >>2A60  
 2A5D YES, DUMMY UP A PHONY BITMAP BLOCK >>038C  
 2A60 ELSE, NORMAL READ/WRITE >>0342

ProDOS Relocator -- V1.1.1 -- 18 SEP 84 NEXT OBJECT ADDR: 2A60  
 ADDR DESCRIPTION/CONTENTS

\*\*\*\*\* READ/WRITE IN AUX HIGH RAM \*\*\*\*\*  
 2AC3 SAVE PAGE NUMBER  
 2AC4 FIND IT IN MEMORY <02E5>  
 2AC7 REMEMBER READ/WRITE STATUS  
 2AC8 WRITING? >>2AB8  
 2ACA GET SAVED PAGE NUMBER  
 2ACB DOES OPERATION INVOLVE BANK1?  
 2ACD NO, USE BANK2 >>2A73  
 2ACF YES, FORCE IT TO \$DXXX  
 2A71 AND USE BANK1 OF AUX HIGH RAM >>2A79  
 2A73 USE BANK2 OF AUX HIGH RAM (C083)  
 2A76 AND WRITE ENABLE IT (C083)  
 2A79 SAVE PAGE NUMBER IN BLOCK (03C1)  
 2A7C PRESERVE HIS BUFFER ADDR (03C0)  
 2A80 DURING THE FOLLOWING TRANSFER... (03BF)  
 2A83 SELECT AUX HIGH RAM (C009)  
 2A88 USE RAMDRIVE BUFFER AS AN "IN BETWEEN" (03C0)  
 2A8B AREA WHEN TRANSFERRING TO/FROM AUX HIGH RAM.  
 2A8D PRETEND THAT WAS CALLER'S BUFFER (03BF)  
 2A90 AND SET UP POINTERS AGAIN <02E5>  
 2A94 COPY BLOCK TO OR FROM RAMDRIVE BUFFER  
 2A9F THEN BACK TO MAIN ZERO PAGE (C008)  
 2AA2 RESTORE CALLER'S BUFFER ADDRESS (03BF)  
 2AA9 READING OR WRITING?  
 2AAA IF WRITING, DONE >>2AB5  
 2AAC IF READING, WRITE ENABLE HIGH RAM (BANK1) (C08B)  
 2AB2 AND COPY RAMDRIVE BUFFER TO HIS BUFFER <02BE>  
 2AB5 THEN EXIT >>03DE  
 2AB8 IF WRITING, COPY HIS BLOCK TO RAMDRIVE BUFFER <02BE>  
 2ABB THEN COPY RAMDRIVE BUFFER TO AUX HIGH RAM >>026A

\*\*\*\*\* COPY BLOCK IN MAIN 48K \*\*\*\*\*  
 2ABE THIS ENTRY IS FOR THE RAMDRIVE BUFFER  
 2AC0 THIS ENTRY ASSUMES AUX MEM PAGE NUMBER IN ACCUM (03C1)  
 2AC3 THIS ENTRY ASSUMES PAGE NUMBER ALREADY SET <02E5>  
 2AC6 WRITING TO RAMDISK? >>2ADB  
 2AC8 NO, WRITE TO MAIN 48K RAM (C004)  
 2ACC COPY BLOCK AUX MEM --> MAIN MEM  
 2AD7 WRITE TO AUX MEM AGAIN (C005)  
 2ADA DONE (RETURN HERE AFTER FOLLOWING JUMP)  
 2ADB ---  
 2ADD GO BACK TO MAIN MEM PART OF DRIVER (03ED)  
 2AE0 TO COPY MAIN MEM --> AUX MEM

ProDOS Relocator -- V1.1.1 -- 18 SEP 84 NEXT OBJECT ADDR: 2AE2

ProDOS Relocator -- V1.1.1 -- 18 SEP 84 NEXT OBJECT ADDR: 2AE2

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

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

2B5B ELSE, FOR BLOCKS \$8 THRU \$5C  
 2B5C SUBTRACT 8  
 2B5E AND DIVIDE BY 17 (\$11)  
 2B64 XREG IS QUOTIENT  
 2B65 HAS TO BRANCH!! >>2B5E  
 2B68 AND AREG IS REMAINDER  
 2B69 REMAINDER OF 1?  
 2B6B NO >>2B73  
 2B6D YES, EVERY 17TH BLOCK GOES  
 2B6E IN \$1000-\$1BFF AREA  
 2B6F BY ADDING 8 TO QUOTIENT  
 2B71 AND GO DO IT >>2B85  
 2B73 BUMP QUOTIENT (START AT \$2XXX)  
 2B75 SHIFT IT TO TOP NIBBLE OF BYTE  
 2B7D GOT A REMAINDER? >>2B81  
 2B7F IF SO, DECREMENT IT (NOT USING 1)  
 2B81 THEN ADD INTO TOP NIBBLE  
 2B82 TO FORM \$10 THRU \$5F (03C1)  
 2B85 BLOCK\*2 FOR PAGE NUMBER  
 2B86 COPY THE BLOCK <02C0>  
 2B89 THEN EXIT >>03DE

2AE5 \*\*\*\*\* SET BUFFER AND BLOCK ADDRESSES \*\*\*\*\*  
 2AE8 GET COMMAND (03BD)  
 2AE8 READ OR WRITE?  
 2AE9 WRITE? >>2B08  
 2AEB NO, GET HIGH BYTE OF BUFFER TO BE READ (03CW)  
 2AF2 AND LOW BYTE OF BUFF ADDRESS (03BF)  
 2AF5 \$42/43 --> FIRST PAGE OF BUFFER  
 2AF7 \$40/41 --> SECOND PAGE OF BUFFER  
 2AF9 GET PAGE NUMBER (03C1)  
 2AFE \$3C/3D --> BLOCK IN RAMDRIVE  
 2B00 \$3E/3F --> SECOND PAGE OF SAME  
 2B06 ALWAYS BRANCH AROUND WRITE CODE >>2B23  
 2B08 WRITE, (03C0)  
 2B0F \$3C/3D --> MAIN MEMORY ADDRESS OF BUFFER TO BE WRITTEN (03BF)  
 2B12 \$3E/3F --> SECOND PAGE OF SAME  
 2B19 \$42/43 --> BLOCK IN RAMDRIVE  
 2B1B \$40/41 --> SECOND PAGE OF SAME  
 2B23 SET SECOND PAGE ADDRESSES  
 2B27 EXIT

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

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

2B8C USE RAMDRIVE BUFFER (NO ACTUAL BITMAP BLOCK)  
 2B91 SET UP BUFFER POINTERS <02E5>  
 2B94 WRITING? >>2BA9  
 2B96 NO, READING - ZERO THE RAMDRIVE BUFFER <0336>  
 2B9B COPY BITMAP IMAGE TO RAMDRIVE BUFFER (03C2)  
 2BA3 COPY BLOCK BACK TO CALLER'S BUFFER <02C3>  
 2BA6 THEN EXIT >>03DE  
 2BA9 WRITING, COPY HIS BUFFER TO RAMDRIVE BUFFER <02C3>  
 2BAC SET UP BUFFER POINTERS <02E5>  
 2BB1 COPY 16 BITMAP BYTES FROM RAMDRIVE BUFFER  
 2BB3 INTO PAGE 3 BITMAP IMAGE (03C2)  
 2BB9 THEN EXIT >>03DE

2B28 ZERO RAMDRIVE BUFFER IN CASE READING <0331>  
 2B2B COPY BETWEEN RAMDRIVE BUFFER AND HIS BUFFER <02C3>  
 2B2E AND EXIT >>03DE  
 2B31 \*\*\*\*\* ZERO BLOCK BUFFER \*\*\*\*\*  
 2B31 ZERO RAMDRIVE BUFFER  
 2B33 ZERO BLOCK INDICATED BY ACCUM. (03C1)  
 2B36 SET UP BUFFER POINTERS <02E5>  
 2B3A ZERO BOTH PAGES OF BLOCK  
 2B41 AND EXIT  
 2B42 \*\*\*\*\* READ/WRITE IN LOW 48K \*\*\*\*\*  
 2B42 BLOCK 2 (VOLUME DIRECTORY)?  
 2B44 NO >>2B4A  
 2B46 YES, CONVERT IT BLOCK 7  
 2B48 AND GO DO I/O NOW >>2B58  
 2B4A ELSE, LESS THAN BLOCK 8? (BUG--\$D SHOULD BE \$F!!!)  
 2B4C YES, RETURN WITH DUMMY ZERO BLOCK. >>2B28  
 2B4E START MSB AT ZERO  
 2B50 GET ORIGINAL BLOCK NUMBER  
 2B52 BLOCK \$5D THROUGH \$5F?  
 2B54 NO >>2B5B  
 2B56 YES, ADJUST TO \$D THROUGH \$F  
 2B58 AND USE \$1A00 THRU \$1FFF IN RAMDRIVE. >>0385

2BBC \*\*\*\*\* RAM DRIVE DATA (AT \$3BC) \*\*\*\*\*

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

2BBC FIRST TIME ENTRY FLAG  
 2BBD COMMAND FROM PARM LIST  
 2BBE UNIT NUMBER FROM PARM LIST  
 2BBF BUFFER ADDRESS FROM PARM LIST  
 2BC1 BLOCK NUMBER FROM PARM LIST

2B42 BLOCK 2 (VOLUME DIRECTORY)?  
 2B44 NO >>2B4A  
 2B46 YES, CONVERT IT BLOCK 7  
 2B48 AND GO DO I/O NOW >>2B58  
 2B4A ELSE, LESS THAN BLOCK 8? (BUG--\$D SHOULD BE \$F!!!)  
 2B4C YES, RETURN WITH DUMMY ZERO BLOCK. >>2B28  
 2B4E START MSB AT ZERO  
 2B50 GET ORIGINAL BLOCK NUMBER  
 2B52 BLOCK \$5D THROUGH \$5F?  
 2B54 NO >>2B5B  
 2B56 YES, ADJUST TO \$D THROUGH \$F  
 2B58 AND USE \$1A00 THRU \$1FFF IN RAMDRIVE. >>0385

ProDOS Relocator -- V1.1.1 -- 18 SEP 84      NLXT OBJECT ADDR: 2BC1  
 ADDR    DESCRIPTION/CONTENTS

```

2BC2 BIT MAP IMAGE FOR RAM DRIVE
2BD2 RAMDRIVE VOLUME NAME
2BD3 'RAM'
2BD6 ACCESS, ENTRY LENGTH
2BD8 NUMBER OF ENTRIES
2BD9 FILE COUNT
2BDB BIT MAP BLOCK POINTER
2BDD BLOCKS ON DISK

2BDE ***** EXIT TO MAIN MEMORY *****
2BEE WRITE ENABLE HIGH RAM (BANK1) (C08B)
2BE5 RESTORE 80STORE STATUS >>2BEA
2BE7 80STORE WAS ON (C001)
2BEA GO AROUND MEMORY USED BY XFER >>03EF
2BED LOW-ORDER BYTE AND
2BEE HIGH-ORDER BYTE USED BY XFER ROUTINE
2BEF RETURN TO $FF44 (NORMAL EXIT)
2BFF USE ROM XFER ROUTINE TO DO IT >>C314
2BFE TWO BYTES NOT USED

2C00 ***** RAMDRIVE CALLER (RUNS AT $FF00) *****
      (USED TO CALL MAIN PART OF RAMDRIVE DEVICE
      DRIVER WHICH IS AT $200 IN AUX MEMORY.
      ROUTINE AT $FF62 IS USED TO TRANSFER DATA
      FROM MAIN TO AUX MEM.)
2C00 SAVE ZPAGE STUFF I WILL CLOBBER
2C03 FROM $3C THRU $47 (FF81)
2C0D SAVE $3ED/E THAT XFER ROUTINE WILL CLOBBER (03ED)
2C16 COMMAND = STATUS?
2C18 IF SO, SIMPLE EXIT WILL DO >>2C44
2C1A ELSE, TOO BIG A COMMAND NUM?
2C1C IF SO, ERROR >>2C3B
2C1E ELSE, INVERT BITS OF CMD
2C20 AND SAVE IT
2C22 FORMAT? >>2C2C
2C24 NO, CHECK BLOCK NUMBER
2C28 MUST BE <128 FOR RAMDRIVE
2C2C GOING TO $200 IN AUX MEMORY
FF33
2C38 USE XFER ROUTINE TO GET THERE >>C314

2C3B I/O ERROR RETURN CODE
2C3D EXIT >>2C41
2C3F WRITE PROTECTED RETURN CODE
2C41 ---
2C42 ERROR EXIT >>2C47
  
```

ProDOS Relocator -- V1.1.1 -- 18 SEP 84      NEXT OBJECT ADDR: 2C44  
 ADDR    DESCRIPTION/CONTENTS

```

2C44 NORMAL EXIT, RETURN CODE IS 0
2C47 ---
2C4B RESTORE ZERO PAGE (FF81)
2C53 AND $3ED/E (FF7F)
2C61 AND EXIT TO CALLER WHEN THRU

2C62 ***** COPY MAIN TO AUX BLOCK *****
      (CALLED FROM AUX MEM HANDLER)
FF62
2C62 WRITE IN AUX 48K (C005)
2C67 COPY BOTH PAGES OF BLOCK
2C72 WRITE IN MAIN 48K AGAIN (C004)
2C77 GO TO $2DA IN AUX MEMORY TO RETURN (03ED)
2C7C RETURN TO AUX MEM HANDLER AGAIN >>FF33

2C7F ***** DATA AREA *****
FF7F
2C7F SAVE $3ED,$3EE
FF80
FF81
2C81 ZERO PAGE SAVE AREA

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

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

ProDOS MLI -- V1.1.1 -- 18 SEP 84 NEXT OBJECT ADDR: D700

ProDOS MLI -- V1.1.1 -- 18 SEP 84 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 file 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	***** SYSTEM GLOBAL PAGE EQUATES *****
BF00	Jump to MLI entry point
BF03	JSPARE (Jump to \$EECF, QUIT code)
BF06	DATEIME 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
BF31	Count (-1) active devices
BF32	List of active devices by DEVID
BF58	Memory BITMAP for low 48K
BF70	Open file 1 buffer address

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 1.1.1 -- 18 SEP 84 *
*	*****
D700	***** ZERO PAGE USAGE *****
0040	Pointer to callers parmlist
0041	-- device driver parmlist --
0042	Command
0043	Unit Number
0044	Buffer Pointer
0045	
0046	Block Number
0047	
0048	I/O Pointer - Index Block or..
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 -- V1.1.1 -- 18 SEP 84 NEXT OBJECT ADDR: D700  
 ADDR DESCRIPTION/CONTENTS

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  
 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 \*\*\*\*\* SOFT SWITCHES \*\*\*\*\*  
 C00C Reset 80 column mode  
 C051 Set TEXT mode  
 C053 Set Mixed text/graphics  
 C054 Display Primary page  
 C056 Set LORES graphics mode  
 CFFF 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.  
 D700 pathname buffer  
 D800 \*\*\*\*\* FILE CONTROL BLOCKS \*\*\*\*\*  
 D800 Reference Number  
 FCB0 starts here..

ProDOS MLI -- V1.1.1 -- 18 SEP 84 NEXT OBJECT ADDR: D800  
 ADDR DESCRIPTION/CONTENTS

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  
 IXXX 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 XXIX Allocate new Sub Index Block

D808 XXXX XXXI Allocate new Data Block  
 D809 Access Byte  
 D80A Newline Character  
 D80B 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  
 D900 \*\*\*\*\* VOLUME CONTROL BLOCKS \*\*\*\*\*

VCB0 starts here..  
 D900 Length (0000LLLL)  
 D901 File Name (Max 15)  
 D910 Unit Number  
 D911 Files Open Flag (if \$FF)  
 D912 Total Blocks  
 D914 Blocks Free  
 D916 Block Number of Vol Dir Key Block  
 D918 not used  
 D919 not used  
 D91A Bit Map Pointer  
 Block offset into multi-block bitmap of  
 D91C next free bit.  
 D91E Count of open files



PRODOS MLI -- V1.1.1 -- 18 SEP 84      NEXT OBJECT ADDR: D91E  
 ADDR    DESCRIPTION/CONTENTS

D920    VCBI through VCB7  
 DA00    \*\*\*\*\* BITMAP BUFFER \*\*\*\*\*  
 DA00    Buffer 1st half  
 DB00    Buffer 2nd half  
 DC00    \*\*\*\*\* PRIMARY BUFFER \*\*\*\*\*  
       (Used for several things. VOL DIR HDR is mapped into it below)

DC00    Pointer Fields  
       \*\*\* VOLUME DIRECTORY HEADER \*\*\*  
 DC04    Type/Length (TTTTLLL)  
 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    Total Blocks  
 DC2B    (remainder of first page of block)  
 DD00    (second page of block)  
 DE00    \*\*\*\*\* MLI MAIN ENTRY POINT \*\*\*\*\*

DE00    Clear decimal mode  
 DE01    Save Registers (BF9F)  
 DE07    Set (\$40) -> Address of function code -1  
 DE0B    Set CMDADR -> True return address  
 DE1A    Init Global Page System error to 0 (BF0F)  
 DE1E    Get Function Code  
 DE21    Build hash index into Command Table (X reg)  
 DE2A    Is this code valid?  
 DE2F    NO >>DEA7  
 DE32    Set (\$40) -> Parameter list  
 DE3F    Get parameter count required (FD85)  
 DE42    None? >>DE60  
 DE44    NO - is parameter count correct?  
 DE46    NO >>DEAB  
 DE48    Check class of function (FD65)  
 DE4B    Quit?  
 DE4D    Yes >>DE5D  
 DE4F    no,  
 DE50    \$8X - Calls to I/O Drivers >>DE66  
 DE52    \$CX/DX - Non System calls >>DE71

PRODOS MLI -- V1.1.1 -- 18 SEP 84      NEXT OBJECT ADDR: DE54  
 ADDR    DESCRIPTION/CONTENTS

DE54    Else, \$4X - Interrupt support  
 DE55    Isolate type (DEALLOC = 1, ALLOC = 0)  
 DE57    Call Interrupt Support <DEF3>  
 DE5A    Then Exit to Caller >>DE78  
 DE5D    Go to quit code via global page >>BF03  
 DE60    \*\*\*\*\* MLI GET\_TIME CALL \*\*\*\*\*

DE60    Call Date/Time driver <BF06>  
 DE63    and exit to caller >>DE78  
 DE66    \*\*\*\*\* MLI READ BLOCK CALL \*\*\*\*\*  
       \*\*\*\*\* MLI WRITE BLOCK CALL \*\*\*\*\*  
       \$80 - Read Block  
       \$81 - Write Block

DE66    ---  
 DE67    Set \$42 -> 1 for READ, 2 for WRITE  
 DE6B    Do Block I/O <DEB2>  
 DE6E    Then Exit to Caller >>DE78  
 DE71    \*\*\*\*\* \$CX and \$DX CALLS \*\*\*\*\*  
 DE72    ---  
 DE75    Isolate function Index  
 DE75    Perform function and exit to caller <E047>

DE78    \*\*\*\*\* EXIT TO CALLER \*\*\*\*\*  
 DE78    Clear Backup  
 DE80    Error occurred?  
 DE83    Save test results  
 DE84    Disable interrupts  
 DE85    MLI no longer active (BF9B)  
 DE88    Get test results back  
 DE89    Store in X reg  
 DE8A    Set up Return Address on stack (BF9D)  
 DE92    Put test results on stack  
 DE94    Put error code in A reg  
 DE95    Restore X reg (BF9E)  
 DE98    Restore Y reg (BF9F)  
 DE9B    Put error code on stack  
 DE9C    Get RAM/ROM orientation (BFF4)  
 DE9F    Exit via RAM Global Page >>BFA0

```

ProDOS MLI -- V1.1.1 -- 18 SEP 84      NEXT OBJECT ADDR: DE9F
ADDR  DESCRIPTION/CONTENTS
-----

```

```

DEA2 ***** NO DEVICE CONNECTED *****

```

```

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

```

```

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

```

```

DEA7 ---
DEA9 Branch always taken >>DEAD

```

```

DEAB ***** BAD PARAMETER COUNT *****

```

```

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

```

```

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

```

```

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

```

```

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

```

```

DEDA ---
DEDC Force off unused UNIT bits
DEE3 Put Drive number in X reg
DEE7 Put Device Handler Address in Jump Vector (FEF5)
DEF0 Exit through Device Handler >>FEF5

```

```

DEF3 ***** Interrupt Handler *****
      ALLOC/DEALLOC

```

```

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

```

```

ProDOS MLI -- V1.1.1 -- 18 SEP 84      NEXT OBJECT ADDR: DEF6
ADDR  DESCRIPTION/CONTENTS
-----

```

```

ALLOC

```

```

DEF8 ---
DEFA Look for empty slot (BF7E)
DF01 His Address better be non-zero
DF05 Store Address of His routine in Global Page (BF7E)
DF0E And return the position number we used
DF14 Exit
DF15 Skip this Vector
DF17 Last one?
DF19 No, check another >>DEFA
DF1B Yes, Table Full Error
DF1D Always taken >>DF21
DF1F Bad Parameter Error
DF21 Call System Error Handler <BF09>

```

```

DEALLOC

```

```

DF24 ---
DF26 Get Position Number
DF28 Can't be zero >>DF1F
DF2C Or greater than 4 >>DF1F
DF2F Make Index into Table from it
DF32 And zero His Vector (BF7E)
DF39 Then Exit

```

```

DF3A ***** IRQ Handler *****

```

```

DF3A ---
DF3C Save A reg from Monitor (BF88)
DF3F And X,Y,S and P (BF89)
DF49 Is this ROM enhanced? (DFD8)
DF4C Yes, skip three pulls >>DF5A
DF53 And RTI Address (BF8E)
DF5A Replace stack to original condition
DF5E Save active slot index (DFCE)
DF61 In bottom half of stack?
DF64 Yes, pop off 16 bytes and save them
DF66 ---
DF6D Save $FA - $FF (top of zero page)
DF6F ---
DF77 Is there a User Vector #1 (BF81)
DF7A No >>DF81
DF7C Yes, call it <DFD9>
DF7F His interrupt? >>DEFA4
DF81 Is there a User Vector #2 (BF83)
DF84 No >>DF8B
DF86 Yes, call it <DFDC>
DF89 His interrupt? >>DEFA4
DF8B Is there a User Vector #3 (BF85)

```

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

NEXT OBJECT ADDR: DF8E

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

NEXT OBJECT ADDR: E044

ADDR DESCRIPTION/CONTENTS

ADDR DESCRIPTION/CONTENTS

```
DF8E No >>DF95
DF90 Yes, call it <DFDF>
DF93 His interrupt? >>DFA4
DF95 Is there a User Vector #4 (BF87)
DF98 No >>DF9F
DF9A Yes, call it <DFE2>
DF9D His interrupt? >>DFA4
DF9F Indicate error type 1
DFA1 Call System Death Handler <BF0C>
DFA4 Interrupt Serviced
DFA6 Restore zero page (FDF5)
DFAE And stack (BF8B)
DFBE Is this enhanced ROM? (DFD8)
DFC1 Yes, skip some stuff we used to have to do >>DFD5
DFC3 Reload X and Y (BF8A)
DFC9 Disable I/O ROMS (CFFF)
DFCC Replace active slot number (C100)
DFD5 Exit from Interrupt >>BFD0
DFD8 ENHANCE FLAG. Set to 1 by RELOCATOR if new type ROM found.
      (That is, if ROM IRQ Vector jumps below $D000)
DFD9 User Interrupt Handlers (#1 - #4) >>BF80
```

DFE5 \*\*\*\*\* SYSTEM ERROR HANDLER \*\*\*\*\*

```
DFE5 Save Error Code (BF0F)
DFE9 Pop out of subroutine
DFEA Exit to caller with Error Code (BF0F)
DFEE RETURN
```

DFEF \*\*\*\*\* SYSTEM DEATH HANDLER \*\*\*\*\*

```
DFEF ---
DFE1 Entry from System Global Page here
DFE2 Turn off 80 column card (C00C)
DFE5 Select standard Text display (C051)
E001 Blank out a line
E003 ---
E008 Print "INSERT SYSTEM DISK AND RESTART" (FE1E)
E012 Go into infinite loop if no error code >>E044
E016 "-" (07F1)
E01B "E" (07F2)
E020 "R" (07F3)
E023 "R" (07F4)
E027 Convert error code to Hex
E033 And print it (07F6)
E037 Second digit also
E044 Infinite loop >>E044
```

E047 \*\*\*\*\* PERFORM FILING OR \*\*\*\*\*  
 \*\*\*\*\* HOUSEKEEPING FUNCTIONS \*\*\*\*\*

```
E047 Save function index (FEB7)
E04A Get INFO flags for this command (FD0D)
E04D Times 2
E04E Store Command Number times 2 (FEB3)
E053 And use it to index into Address Table
E057 Set up Jump Vector with this function's (FEF5)
E05A ..handler address (FDA6)
E060 Signal Backup required after call
E065 PATHNAME not required? >>E06C
E067 Required - parse and validity check <E08A>
E06A Bad Name? >>E083
E06C Reference Number in list? (FEB3)
E06F No >>E076
E071 Yes - check it out <ELD0>
E074 Bad Number? >>E083
E076 Date/Time in list? (FEB3)
E079 No >>E07E
E07B Yes - set System date just in case <BF06>
E07E Call Function Handler <E087>
E081 If no errors then exit >>E086
E083 Else - call System error handler <BF09>
E086 Return to caller
E087 Indirect JUMP to Handler >>FEF5
```

E08A \*\*\*\*\* CHECK CALLER'S PATHNAME \*\*\*\*\*  
 \*\*\*\*\* COPY TO MY AREA \*\*\*\*\*

```
E08A Set ($48) -> Pathname
E095 ---
E099 Assume partial Pathname (FEB3)
E09C No Pathname in my area yet (D700)
E09F Check length of caller's Pathname
E0A1 Zero is no good >>E0FB
E0A5 Nor is 65 or more >>E0FB
E0A7 Save length (FE9E)
E0AA Length + 1 (FE9E)
E0AE Get first character of his name
E0B2 Is it "/"?
E0B4 No >>E0BA
E0B6 Yes - indicate fully qualified name (FEB3)
E0B9 Bump past "/"
E0BA ---
E0BC Length of Index level is -1 initially (D700)
E0BF First character of Index level (counter) (FEB8)
E0C2 Start of upcoming Index level in name (FEBA)
E0C5 At end of name yet? (FE9E)
E0C8 Yes >>E0FF
```

PRODOS MLI -- V1.1.1 -- 18 SEP 84 NEXT OBJECT ADDR: E0CA  
 ADDR DESCRIPTION/CONTENTS

E0CA No - get next character in his name  
 E0D0 Is it "/"?  
 E0D2 Yes >>E114  
 E0D4 No - lower case?  
 E0D6 No >>E0DA  
 E0D8 Yes - force upper case  
 E0DA Copy to my Pathname buffer (D703)  
 E0DD Increment Index level counter (FE88)  
 E0E0 Subsequent characters may be A-Z,0-9 or . >>E0E7  
 E0E2 Increment index level counter (FE88)  
 E0E5 First character must be alphabetic >>E0F3  
 E0E7 Is it "."?  
 E0E9 Yes - get next character >>E0C5  
 E0EB No - is it special or control character  
 E0ED Yes - Bad Pathname then >>E0FB  
 E0EF Is it numeric?  
 E0F1 Yes - get next character >>E0C5  
 E0F3 Is it Alphabetic?  
 E0F9 If so get next character >>E0C5  
 E0FB Else  
 E0FC Bad Pathname  
 E0FE RETURN  
 E0FF ---  
 E101 Any characters in last Index level? (FE88)  
 E104 Yes >>E10A  
 E106 No, zero characters in it (FE88)  
 E109 And toss out last "/"  
 E10A ---  
 E10B Mark end of name with \$00 (D700)  
 E10E Name too long? >>E0FB  
 E110 No - save final length (FE9E)  
 E113 Set X -> 0  
 E117 Last Index more than 15 characters?  
 E119 Yes - then no good >>E0FB  
 E11B Save output Index (FE8D)  
 E11E Store length of previous Index level (FE8A)  
 E121 Just before it in buffer (D700)  
 E124 Restore output index (FE8D)  
 E127 And continue >>E0BA  
 E129 End of Name  
 E12A Fully qualified name? (FEBC)  
 E12D Yes >>E134  
 E12F No - Got a Prefix (BF9A)  
 E132 NO - error >>E0FB  
 E134 Else, okay to exit

PRODOS MLI -- V1.1.1 -- 18 SEP 84 NEXT OBJECT ADDR: E135  
 ADDR DESCRIPTION/CONTENTS

E135 \*\*\*\*\* MLI SET PREFIX CALL \*\*\*\*\*  
 \*\*\*\*\* MLI SET PREFIX CALL \*\*\*\*\*  
 E135 Copy Pathname <E08A>  
 E138 It's okay >>E144  
 E13A Check length of Volume name (D700)  
 E13F If zero - no Prefix wanted (BF9A)  
 E142 Exit with no error  
 E143 RETURN  
 E144 Get File entry for last index <E5A3>  
 E147 Okay? >>E144  
 E149 Invalid Pathname?  
 E14B No - Out now! >>E18B  
 E14D Sub Directory file? (FE5F)  
 E154 No, error >>E189  
 E156 Fully qualified path? (FEBC)  
 E159 Yes >>E15E  
 E15B No - use old Prefix also (BF9A)  
 E15E ---  
 E160 Compute new Prefix Index (FE9E)  
 E163 Does new Prefix exceed 64 characters?  
 E165 Yes - Bad Path error >>E0FB  
 E168 Store new Prefix pointer (BF9A)  
 E16E Set Device Number of Prefix Directory (FE9F)  
 E174 Save Keyblock for Prefix Directory (FEA0)  
 E17D Copy Prefix to top of Path buffer (D700)  
 E180 (preceded by old Prefix if one exists) (D700)  
 E188 Exit normally  
 E189 Bad File Type Error  
 E18B ---  
 E18C RETURN  
 E18D \*\*\*\*\* MLI GET PREFIX CALL \*\*\*\*\*  
 \*\*\*\*\* MLI GET PREFIX CALL \*\*\*\*\*  
 E18D Set (\$4E) -> Data Buffer  
 E199 Set Length = 64 (max)  
 E1A3 Validity check buffer storage <FC82>  
 E1A6 Get Prefix? >>E18B  
 E1AA Get Prefix index (BF9A)  
 E1AE No Prefix? - Length = 0 >>E1B4  
 E1B0 Complement for length  
 E1B4 Store in first byte of buffer  
 E1B6 If null Prefix exit >>E1CE  
 E1B8 ---  
 E1B9 Copy Prefix to caller's buffer replacing (D700)  
 E1BC index level name length bytes with "/"

ProDOS MLI -- V1.1.1 -- 18 SEP 84      NEXT OBJECT ADDR: E1C6  
 ADDR    DESCRIPTION/CONTENTS

---  
 E1C6 End it with a "/"  
 ---  
 E1CE  
 E1CF Exit normally

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

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

---  
 E22C This looks wrong!!!! (FE92)  
 E231 Bad Reference Number error  
 E234 RETURN

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

ProDOS MLI -- V1.1.1 -- 18 SEP 84      NEXT OBJECT ADDR: E235  
 ADDR    DESCRIPTION/CONTENTS

E235 Set (\$4E) -> Data Buffer <F20D>  
 E238 Set Length = 0  
 E242 Get Unit Number  
 E244 Do all Units? >>E24D  
 E246 No, just one  
 E248 Set length = 16 (FEDA)  
 E24B Always taken >>E252  
 E24D If all Units  
 E24F Set Length = 256 (maximum) (FEDB)  
 E252 Is Buffer in main RAM? <FC82>  
 E255 No, then exit >>E28A  
 E257 Yes, zero out Buffer  
 E25C ---  
 E261 Index into Data Buffer = \$00 (FEBA)  
 E266 Get Unit Number again  
 E268 Isolate valid bits  
 E26A Specific Unit requested? >>E28B  
 E26C No, copy Device List from Global Page <E864>  
 E26F Save Device Count (FEBD)  
 E272 Get last Device (FECA)  
 E275 Generate return data for it <E28B>  
 E278 Bump data buffer index by 16 (FEBA)  
 E281 Get next Device (FEBD)  
 E285 And go do it >>E26F  
 E287 When done, exit  
 E28A RETURN  
 E28B Save Device Number (BF30)  
 E28E Scan for the Volume Control Block <E876>  
 E291 Error? >>E2C3  
 E293 We need Block 2 (Key Block of VolDir)  
 E29B Read Volume Directory Key Block <EBEE>  
 E29E Error? >>E2C3  
 E2A0 Was something already mounted? (FE91)  
 E2A6 No >>E2AD  
 E2A8 Yes, Files open? (D911)  
 E2AB Yes >>E2B9  
 E2AD No, set up Volume Control Block for new VOL <E8D1>  
 E2B0 Error? >>E2C3  
 E2B2 No  
 E2B4 Was a duplicate Volume Control Block found? (FE95)  
 E2B7 Yes, then error >>E2C3  
 E2B9 See if the same Volume is still there (FE91)  
 E2BF If not, Disk Switch Error  
 E2C1 Else, all is well - continue >>E2E1

```

ProDOS MLI -- V1.1.1 -- 18 SEP 84          NEXT OBJECT ADDR: E2C3
-----
ADDR  DESCRIPTION/CONTENTS
-----

```

```

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

```

```

E2C3 ---
E2C4 Store Device Number in entry <E2F6>
E2C9 Store error code next
E2CB Duplicate Volume error?
E2CD No - done >>E2DF
E2D0 Store Device Number for duplicate next (FEB6)
E2D8 NO Duplicate now
E2DF Exit with error
E2E0 RETURN

```

```

E2E1 ***** MAKE ONLINE VOLUME ENTRY *****
E2E1 Get name length for loop index (D900)
E2EA Copy name to Buffer entry (D900)
E2F1 Done yet? (FEB8)
E2F4 No, do another >>E2EA
E2F6 Yes, find current Buffer entry (FEBA)
E2F9 Store Device number (BF30)
E301 Return to caller

```

```

E302 ***** MLI CREATE CALL *****
      *****

```

```

E302 Follow Path to File <E5B6>
E305 Error? - I'm expecting one >>E30B
E307 If File was found - Duplicate error
E309 ---
E30A Return to caller

E30B File not found?
E30D No, then a real error occurred >>E309
E30F Yes, get requested storage type
E313 Is it 00, $01, $02 or $03?
E315 Yes, carry on >>E31B
E317 Is it $0D?
E319 NO, then exit with error >>E32B
E31B Get status of this device (BF30)
E321 Exit on error >>E32E
E323 Is there a free Directory entry? (FE9B)
E326 No >>E32F
E328 Yes - continue >>E3C1

```

```

ProDOS MLI -- V1.1.1 -- 18 SEP 84          NEXT OBJECT ADDR: E328
-----
ADDR  DESCRIPTION/CONTENTS
-----

```

```

E32B Indicate Bad Storage Type
E32E Return to caller

```

```

E32F Is this the Volume Directory? (FE46)
E335 No, we can extgnd it >>E33B
E337 Yes, indicate Volume Directory Full error
E33A Return to caller

```

```
* EXTEND DIRECTORY FILE *
```

```

E33B Save old current Block number
E341 Allocate a Block on Disk <EAB6>
E344 Save the number
E345 Replace BLKNUM
E34B Was there a free Block?
E34C No, then exit >>E32E
E34E Yes, set up forward pointer in old one (DC02)
E351 to point to it (DC03)
E354 and Write old Directory Block <EBEA>
E357 Error? Yes, then exit >>E32E
E35B Set BLKNUM -> new Block number
E360 Back point to old Directory Block (DC02)
E366 Loop until done >>E35B
E36A Zero remainder of Block Buffer (DC02)
E36D (including forward pointer) (DB00)
E371 Loop until done >>E36A
E373 Write new Directory Block <EBEA>
E376 Error? Yes, then exit >>E32E
E378 Set BLKNUM -> Parent Directory number (FE46)
E382 Read Block with my entry <EBEE>
E385 Entry number of my Directory (FE48)
E388 None relocatable!!
E38A Set ($48) -> Buffer
E38C Skip link pointers
     ---
E38E Count entries
E38F Skip to next (FE49)
E392 Save LSB
E39B Add 1 to Blocks used
E39F and $200 to EOF mark (FDCE)
E3A1 in entry
E3A4 Loop until done >>E39F
E3AA Write back Block to Parent Directory <EBEA>
E3AC Error? then exit >>E3C0
E3AF Start all over now that there's room >>E302
E3B1

```

PRODOS MLI -- V1.1.1 -- 18 SEP 84 NEXT OBJECT ADDR: E3B1  
 ADDR DESCRIPTION/CONTENTS

E3B4 \*\*\*\*\* ZERO \$F600 \*\*\*\*\*  
 E3B4 Zero \$F600 Block Buffer  
 E3C0 Return to caller  
 E3C1 \*\*\*\*\* BUILD NEW FILE \*\*\*\*\*  
 E3C1 Call Zero \$F600 routine <E3B4>  
 E3C4 Copy Datetime (Creation)  
 E3C6 to my variables  
 E3D2 Loop until done >>E3C6  
 E3D4 bid he give Datetime (Creation)?  
 E3D5 Yes, carry on >>E3E2  
 E3D7 No, then use  
 E3D9 System Datetime instead (BF90)  
 E3E2 If Storage type is \$00, \$01, \$02 or \$03  
 E3E4 force it to \$10  
 E3EA else use a \$D0  
 E3EC Find File name (FEBA)  
 E3EF OR Storage type to name length (D700)  
 E3F2 Store Type/Length (FE5F)  
 E3F5 Isolate name length  
 E3F9 Copy File name to File Entry Buffer (FEBA)  
 E407 Copy caller's Access Byte  
 NOTE: This should be validity checked!!!  
 and copy File type  
 E40F ---  
 E414 and AUX TYPE  
 E415 Copy Version and Min Version (0,0) (FDF0)  
 E41E constants to entry (FE7B)  
 E421 Indicate 1 Block used  
 E42F Copy Directory Header Block number (FE5A)  
 E43E Is this a Seedling file?  
 E440 Yes >>E479  
 E442 No, Directory file - Build Header in \$F600  
 E444 Copy completed Directory entry (FE5F)  
 E447 to \$F600 buffer first (DC04)  
 E44B Loop until done >>E444  
 E44D Make Storage type \$E in Header itself  
 E452 Put "HUSTON" (Author) in Reserved area  
 E45A and Version, Min Version, Access, (FDF0)  
 E45D Entry-length, File count and (DC20)  
 E460 Parent pointer from constants  
 E461 Loop until done >>E454  
 E465 Copy Parent Block entry number (FE5C)  
 E46C Loop until done >>E465  
 E46E Copy Parent entry Length (FE51)  
 E476 EOF = \$200 (FE75)  
 E479 Allocate a new disk block <EAB6>  
 E47C error? >>E4B5

PRODOS MLI -- V1.1.1 -- 18 SEP 84 NEXT OBJECT ADDR: E47E  
 ADDR DESCRIPTION/CONTENTS

E47E Store it in key pointer of entry (FE70)  
 E484 and in BLKNUM for I/O  
 E488 Write zeroed (or DIR HDR) key block <EBEA>  
 E48B error? >>E4B5  
 E48D Bump parent's file count (FE53)  
 E495 Go update directory <E4B6>  
 E498 error? >>E4B5  
 E49A Checkpoint Volume Bit Map and exit. >>EB93  
 E49D \*\*\*\*\* POINT \$48/49 AT DIRECTORY ENTRY \*\*\*\*\*  
 E49D \$48/\$49 --> Entry  
 E4A1 Skip link pointers (+4)  
 E4A3 File entry number counter (FE5E)  
 E4A6 ---  
 E4A7 Skip to proper entry  
 E4AA Add entry length (FE51)  
 E4AF (bump MSB)  
 E4B3 (store LSB)  
 E4B5 RETURN  
 E4B6 \*\*\*\*\* UPDATE DIRECTORy(S) \*\*\*\*\*  
 E4B6 System date available? (BF90)  
 E4B9 no, forget it >>E4C6  
 E4BD yes, copy to last modified date field (BF90)  
 E4C6 turn on BUBIT (backup) if appropriate (FE7D)  
 E4CF set DEVNUM of parent (FE59)  
 E4D5 and BLKNUM (FE5C)  
 E4DF reread DIR block containing entry <EBEE>  
 E4E2 error? >>E4B5  
 E4E4 Point to proper entry in buffer <E49D>  
 E4EB Copy constructed entry to buffer (FE5F)  
 E4F6 Is this block the DIR HDR block?  
 E501 no, write back new entry <EBEA>  
 E504 error? >>E4B5  
 E510 and then read DIR HDR block <EBEE>  
 E513 error? >>E4B5  
 E515 in any case..  
 E517 copy back update file count to HDR (FE53)  
 E520 and ACCESS byte (with Backup) (FE50)  
 E526 write back HDR block <EBEA>  
 E529 error? >>E583  
 E52B is this the VOL DIR? (DC04)  
 E532 Yes, all done -- exit >>E5A1  
 E534 no, subdirectory.. (DC27)  
 E537 get parent pointer  
 E53E get parent entry no.. (DC29)  
 E544 and entry len (DC2A)  
 E54A read parent DIR block <EBEE>  
 E54D error? >>E583

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

NEXT OBJECT ADDR: E54F

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

NEXT OBJECT ADDR: E5DD

ADDR DESCRIPTION/CONTENTS

ADDR DESCRIPTION/CONTENTS

E54F find entry for this subdirectory <B49D>  
 E552 system date available? (BF90)  
 E555 no >>E564  
 E557 Yes,  
 E55B copy system date/time to... (BF90)  
 E55E modified date/time in entry  
 E564 write it back <EBEA>  
 E567 error? >>E583  
 E56B BLKNUM = HDR block number  
 E574 same block we have now?  
 E578 Yes, go back and date stamp >>E52B  
 E57A no,  
 E57E read HDR block <EBEE>  
 E581 and go back to date stamp parent DIR >>E52B  
 E583 error? then exit

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

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

E5E5 indicate no free entry found as yet  
 E5EA signal in HDR block  
 E5EB zero count of names examined  
 E5F0 find name in block <E6E3>  
 E5F3 got it! >>E65A  
 E5F5 not yet, how many entries expected? (FE98)  
 E5F8 less entry no. I just searched (FE97)  
 E5FD more file entries left to search? >>E60F  
 E60B no, directory error  
 E60D ---  
 E60E RETURN

E584 \*\*\*\*\* NOT ProDOS VOLUME ERROR \*\*\*\*\*

E584 ---  
 E587 RETURN

E588 \*\*\*\*\* IS THIS ProDOS VOLUME? \*\*\*\*\*

E588 Does previous block ptr = 0? (DC00)  
 E596 no, not a ProDOS volume >>E584  
 E598 else, (DC04)  
 E59D does VOL DIR's STORAGE TYPE = \$E or \$F?  
 E59F no, error >>E584  
 E5A1 else, ok  
 E5A2 RETURN

E5A3 \*\*\*\*\* GET FILE ENTRY \*\*\*\*\*

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

E5B6 \*\*\*\*\* FOLLOW PATH TO A FILE \*\*\*\*\*

E5B6 get base dir's data <E73A>  
 E5B9 error? >>E60D  
 E5BB another subdirectory in the path? >>E5E5  
 E5BD no, at end of path (E635)  
 E5C0 \$4B/\$49 --> \$F604 (HDR)  
 E5C8 copy part of HDR to file entry  
 E5D2 File type = \$F (Directory) (FDE8)  
 E5D5 BLOCK = 2 (FE5F)  
 E5D8 No. blocks used = 4  
 E5D9 EOF = \$800

E62E free entry found in directory? (FE9B)  
 E631 Yes >>E64E  
 E633 no, check pointers (DC02)  
 E636 is there another block after this one? >>E63D  
 E63B no... >>E64E  
 E63D Yes, free entry will be.. (FE5C)  
 E646 first in that block  
 E64B indicate free entry available (FE9B)  
 E64E find next index name <E77B>  
 E651 exiting with error  
 E652 no more indices in path, file not found >>E657  
 E654 else, path not found  
 E656 RETURN

\*\*\* NO MORE FILE ENTRIES \*\*\*

E657 file not found error  
 E659 RETURN

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



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

ADDR DESCRIPTION/CONTENTS

E65A advance to next subdir in path <E774>  
 E65D end -- save entry no. and exit >>E6CB  
 E661 get type of entry  
 E665 subdir?  
 E667 no, bad path then >>E651  
 E66B copy key block no...  
 E66D to BLKNUM  
 E670 and to current DIR block no (FE5A)  
 E67A go read key block of subdirectory <EBEE>  
 E67D error? >>E6A3  
 E682 check minimum version (DC21)  
 E68B too new? >>E6A1  
 E696 count bits in reserved field of DIR hdr  
 E697 --- >>E69A  
 E69A ---  
 E69D there must be 5 bits on (normally \$75)  
 E69F (there are) >>E6A5  
 E6A1 or else, incompatible file format  
 E6A3 ---  
 E6A4 RETURN

E6A5 copy DIR HDR <E6AB>  
 E6A8 and go scan for next level >>E5E5

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

E6AB COPY:  
 E6AD CREATION, VERSION, MIN\_VERS, ACCESS, (DC1C)  
 E6B0 ENTRY\_LEN, ENTRIES\_PER\_BLK, FILE\_COUNT (FE4A)  
 E6B6 volume directory? (DC04)  
 E6BD if so, exit now >>E6CA  
 E6C1 else, copy PARENT\_POINTER. (DC27)  
 E6C4 PARENT\_ENTRY\_NO., and PARENT\_ENTRY\_LEN (FE46)  
 E6CA RETURN

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

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

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

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

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

NEXT OBJECT ADDR: E6F8

ADDR DESCRIPTION/CONTENTS

E6F8 Yes >>E707  
 E6FA no, do we need one? (FE9B)  
 E6FD no >>E727  
 E6FF Yes, remember it <E6CB>  
 E702 don't need another one now (FE9B)  
 E705 skip to next entry >>E727  
 E707 get length of name  
 E709 count it (FE97)  
 E70C save it for loop (FE88)  
 E712 same len as we are wanting? (D700)  
 E715 no, skip it >>E727  
 E717 ---  
 E71B compare names (D700)  
 E725 we found it! exit  
 E726 RETURN  
 E727 skip to next entry (FE9A)  
 E72B end of block? if so, exit >>E726  
 E731 bump \$48/\$49 by entry len  
 E738 and go check next >>E6F0

E73A \*\*\*\*\* GET DIRECTORY DATA \*\*\*\*\*

E73A find base directory <E793>  
 E73D error? >>E792  
 E743 zero out my variables (FE46)  
 E749 set up device number (BF30)  
 E74F copy DIR HDR to my variables <E6AB>  
 E758 copy TOTAL BLOCKS from VCB (D912)  
 E75E copy BIT MAP Pointer from VCB (D91A)  
 E764 copy Block No. of this directory (0046)  
 E76A make second copy of file count (FE53)  
 E774 advance to next subdir in path <E77B>  
 E777 and update index (FEBA)  
 E77A RETURN

E77B \*\*\*\*\* ADVANCE TO NEXT DIR NAME \*\*\*\*\*

E77B get this DIR's index (FEBA)  
 E782 add len of name to move index to next name (FEBA)  
 E786 still in prefix portion? >>E78E  
 E788 no, now starting caller's path suffix (BF30)  
 E78B save last DEVNUM accessed (FE9F)  
 E78E return with len of next dir in path (D700)  
 E792 RETURN

ProDOS MLI -- V1.1.1 -- 18 SEP 84      NEXT OBJECT ADDR: E793  
 ADDR    DESCRIPTION/CONTENTS

E793 \*\*\*\*\* FIND BASE DIRECTORY \*\*\*\*\*

```

E793    ---
E795    get old PFIPTTR (BF9A)
E798    fully qualified pathname? (FEBC)
E79B    no >>E79E
E79D    yes, no old PFIPTTR anymore
E79E    save old prefix index (FE9B)
E7A1    DEVNUM=0 (BF30)
E7A4    ---
  
```

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

```

E7A6    scan (D900)
E7A9    got one >>E7B7
E7B0    else, bump to next VCB
E7B4    no mounted vols? remount them >>E808
  
```

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

```

E7B7    store name length (FE88)
E7BA    same name as in pathname? (D700)
E7BD    no -- skip it >>E7AB
E7CB    save VCB index (FE91)
E7CE    DEVNUM = VCB's unit no. (D910)
E7D4    BLOCK = 2 (read VOLDIR if no old PFI)
E7DC    get old prefix index (FE9B)
E7DF    ---
E7E0    accumulate a new index (FEBA)
E7E3    no previous prefix? >>E7F5
E7E6    find last name in prefix (D700)
E7EB    read prefix directory instead of vol dir (FEA0)
E7F5    read block <EBEE>
E7F8    error? >>E800
E7FA    is this the right directory? <E89E>
E7FD    no >>E800
E7FF    yes -- exit!
  
```

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

```

E800    open files? (FE91)
E806    yes, give up now >>E821
E808    else, (FE9B)
E80B    put back old prefix length (FEBA)
E80E    copy DVCLST from global page <E864>
E814    use last device accessed first >>E825
E816    if none, get last in my device table (BF31)
E821    volume not found error
E824    RETURN
  
```

ProDOS MLI -- V1.1.1 -- 18 SEP 84      NEXT OBJECT ADDR: E824  
 ADDR    DESCRIPTION/CONTENTS

```

E825    ---
E828    search for device in device table (FECA)
E830    device not found >>E821
E832    when found, make it active device (BF30)
E837    remove it from table (FECA)
E83A    find its VCB <E876>
E83D    not found? >>E863
E83F    volume mounted there? (FE91)
E845    no >>E84C
E847    yes, open files here? (D911)
E84A    yes, skip it -- get next unit >>E816
E84C    else,
E84E    BLKNUM = 2 (vol dir)
E854    read volume directory <EBEE>
E857    error? >>E816
E859    mount volume on VCB <E8C4>
E85C    error? >>E816
E85E    is this his chosen volume? <E89E>
E861    no, try again >>E816
E863    yes, exit
  
```

E864 \*\*\*\*\* COPY GLOB DEVLST TO MY TABLE \*\*\*\*\*

```

E864    start with last device (BF31)
E867    get a unit number (BF32)
E86C    copy it to device table (FECA)
E872    return count of devices (BF31)
E875    RETURN
  
```

E876 \*\*\*\*\* SCAN VCB'S FOR DEVICE NO. \*\*\*\*\*

```

E876    ---
E87A    scan VCB's for a given device number
E881    not it? >>E888
E883    is it, save VCB index (FE91)
E886    and exit normally
E887    RETURN
E888    else, volume mounted here? (D900)
E88B    yes >>E891
E88E    no, save VCB index to empty unit (FE91)
E891    ---
E893    bump to next VCB
E895    and go look at it >>E87A
E897    not found...
E898    any free entries? if not, error >>E89B
E89A    else, all is well -- return empty VCB
E89B    VCB table full error
E89D    RETURN
  
```

ProDOS MLI -- V1.1.1 -- 18 SEP 84      NEXT OBJECT ADDR: E89D

ADDR    DESCRIPTION/CONTENTS

```

E89E ***** COMPARE DIR NAME WITH PATH LVL *****
E89E    ---
E8A3    check DIR type (DC04)
E8A6    VOL DIR or SUB DIR?
E8A8    neither >>E8B1
E8AA    yes..
E8AC    store len of its name (FEB8)
E8AF    and go on >>E8B6
E8B1    error exit
E8B2    RETURN

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

E8C4    ***** MOUNT NEW VOLUME *****
E8C4    volume mounted? (FE91)
E8CA    no, continue >>E8D1
E8CC    yes, same one as one wanted? <E929>
E8CF    if so exit, else fall thru >>E928

E8D1    ***** SET UP VCB FROM VOLDIR *****
E8D1    zero out VCB
E8DC    is this a ProDOS volume? <E588>
E8DF    no -- exit >>E928
E8E1    duplicate vol in VCB's? <E94A>
E8E4    yes -- exit with that one instead >>E927
E8E6    get new volume's name length (DC04)
E8ED    add to VCB index (FE91)
E8F1    and copy to VCB name field in empty VCB (DC04)
E8FC    store in VCB name len field (D900)
E8FF    copy DEVNUM to VCB unit field (BF30)
E905    copy total blocks to VCB (DC29)
E911    copy block no. of vol dir to VCB
E91B    copy bit map block no. to VCB (DC27)
E927    exit
E928    RETURN

E929    ***** COMPARE VOL NAMES TO MAKE *****
E929    ***** SURE THEY MATCH *****
E929    get length (DC04)
E92E    same in VCB? (D900)
E931    no >>E941
E934    yes, add len to VCB index to point at (FE90)
E937    last char of name in VCB (FE90)

```

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

NEXT OBJECT ADDR: E93E

ADDR    DESCRIPTION/CONTENTS

```

E93E    compare names (D900)
E941    SEC if no match
E948    CLC if match
E949    RETURN

E94A    ***** LOOK FOR DUPLICATE VOL *****
E94A    start with first VCB
E94C    ---
E94D    this VCB has same name? <E929>
E950    no >>E961
E952    yes, files open? (D911)
E955    yes >>E96B
E959    no, mark VCB empty (NAME=0) (D900)
E95C    (UNIT=0) (D910)
E95F    and exit with no error >>E969
E961    else,
E963    bump to next VCB
E967    and loop >>E94C
E969    exit no errors
E96A    RETURN

E96B    save flag (FEB5)
E96E    and VCB index of duplicate vol (FEB6)
E971    exit with error
E972    RETURN

E973    ***** SEE IF A QUANTITY OF FREE *****
E973    ***** BLOCKS IS AVAILABLE ON VOL *****
E973    any free blocks counted in VCB? (FE91)
E97C    yes >>E9D0

E97C    *** COMPUTE VCB FREE BLOCK COUNT ***
E97E    no, how many bit map blocks are there? <EA22>
E981    save it (less 1) (FE9C)
E986    zero scratch (will count free blocks) (FE86)
E98C    no block found yet
E991    checkpoint bit map buffer <EB93>
E994    error? >>E9E4
E999    BLKNUM = bit map pointer (D91A)
E9A3    read block to buffer <EBEE>
E9A6    error? >>E9E4
E9A8    count free blocks marked <E9E5>
E9AB    drop no. remaining to do (FE9C)
E9AE    none left? >>E9B9
E9B0    some, BLKNUM = BLKNUM + 1
E9B6    go process that >>E9A3

```

PRODOS MLI -- V1.1.1 -- 18 SEP 84      NEXT OBJECT ADDR: E9B6  
 ADDR    DESCRIPTION/CONTENTS

E9B9 did we find a free bit? (FE91)  
 E9BF no -- volume full >>E9E1  
 E9C1 save VCB bitmap block offset (D91C)  
 E9C4 save free block count in VCB also (FE87)  
 E9D0 are there enough to satisfy request? (D914)  
 E9DF yes, exit  
 E9E0 RETURN

E9E1 volume full error  
 E9E4 RETURN

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

E9E5 scan through both buffer pages  
 E9EC counting one bits <EA12>  
 E9F7 ---  
 E9FA found free block already? (FE9B)  
 E9FD if so -- done >>EAll  
 E9FF any blocks found yet? (FE86)  
 EA05 no >>EAll  
 EA07 yes, compute total no. of bitmap blocks <EA22>  
 EA0B less number remaining (FE9C)  
 EA0E gives bitmap block with first free bit (FE9B)  
 EA11 exit

EA12 \*\*\*\*\* COUNT ONE BITS IN A BYTE \*\*\*\*\*  
 EA12 shift and...  
 EA15 count bits that are on (FE86)  
 EA1D exit when byte goes to zero  
 EA21 RETURN

EA22 \*\*\*\*\* COMPUTE NO. BITMAP BLKS -1 \*\*\*\*\*  
 EA22 get blocks on vol count (-1) (FE91)  
 EA2E ---  
 EA2F isolate top nibble of block count  
 EA30 for bit map block count  
 EA33 RETURN

EA34 \*\*\*\*\* FREE A BLOCK ON DISK \*\*\*\*\*  
 EA34 save MSB (FE9C)  
 EA37 and LSB  
 EA3B block number passed too big for (D913)  
 EA3E volume size? (FE9C)  
 EA42 yes, error >>EAB2  
 EA45 no, get bit position for block no.  
 EA4B save it (FE9B)  
 EA4F divide block no. by 8 (FE9C)

PRODOS MLI -- V1.1.1 -- 18 SEP 84      NEXT OBJECT ADDR: EA52  
 ADDR    DESCRIPTION/CONTENTS

EA52 giving byte offset as remainder  
 EA5B save byte offset (FEA2)  
 EA5E make quotient/2 into block index (FE9C)  
 EA61 remember which page in that block (FEA4)  
 EA64 read bit map block (after checkpoint) <EB64>  
 EA67 error? >>EAB1

EA69 are we at proper block of bitmap yet? (FEA9)  
 EA6F yes! >>EA87  
 EA71 no -- checkpoint <EB93>  
 EA74 error? >>EAB1  
 EA76 indicate block wanted in VCB (FE9C)  
 EA7F DEVNUM of bitmap (FEA6)  
 EA82 read actual block directly <EBA4>  
 EA85 error? >>EAB1

EA87 get byte offset into page (FEA2)  
 EA8A which page? (FEA4)  
 EA90 page 0? >>EA9A  
 EA92 no, turn bit on in page 1 (DB00)  
 EA98 and continue >>EAA0  
 EA9A turn bit on in page 0 (DA00)  
 EAA0 mark bitmap needs checkpoint  
 EAAB count block freed (FEC2)  
 EAB0 exit normally  
 EAB1 RETURN

EAB2 bad bitmap error  
 EAB5 RETURN

EAB6 \*\*\*\*\* FIND A FREE DISK BLOCK AND \*\*\*\*\*  
 \*\*\*\*\* AND ALLOCATE IT \*\*\*\*\*

EAB6 go read bitmap <EB64>  
 EAB9 error? >>EADE  
 EABB first page 0  
 EAC0 scan 1st page of bitmap for free block(s) (DA00)  
 EAC8 bump tm page 1 of buffer (FEA4)  
 EACB bump page offset (FEA3)  
 EACE scan 2nd page too (DB00)  
 EAD6 bump page (FEA3)  
 EAD9 get next block <EB42>  
 EADC continue >>EABB  
 EADE error exit

EADF save byte index (FEA2)  
 EAE2 shift combination of page no. and (FEA3)  
 EAE5 byte offset left 3 bits to make (FE87)  
 EAE8 room for bit position.  
 EAF7 depending on buffer page ... (FEA4)  
 EAFC reload bit pattern from page 0... (DB00)  
 EB01 or page 1 (DA00)

ProDOS MLI -- V1.1.1 -- 18 SEP 34      NEXT OBJECT ADDR: EB04  
 ADDR    DESCRIPTION/CONTENTS

```

EB04 shift bit pattern, bumping block no. LSB
EB05 until a one bit is found >>EB0A
EB0A then shift it back the way it was
EB0B (with that bit turned off) >>EB0A
EB0D store LSB of block no. (FE86)
EB10 store updated byte back in proper page (FEA4)
EB1D indicate bitmap needs checkpoint
EB25 one less block available in VCB (FE91)
EB3A ---
EB3B return with new block no. (FE86)
EB41 RETURN

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

EB60 disk full error
EB63 RETURN

EB64 ***** READ BITMAP BLOCK *****
EB64 have we read bitmap for this unit yet? (FE91)
EB6D yes >>EB7D
EB6F no, checkpoint bitmap of some other unit <EB93>
EB72 error? >>EB92
EB77 get new bitmap unit no. (D910)
EB7D was bitmap modified? (FEA5)
EB80 yes >>EB87
EB82 no, read it <EBA4>
EB85 error? >>EB92
EB87 save bitmap block offset times 2 (FE91)
EB8A (page number) (D91C)
EB91 exit
EB92 RETURN

EB93 ***** CHECKPOINT VOLUME BITMAP *****
---
EB93 needs checkpoint? (FEA5)
EB94 no >>EB92
EB99 yes, write it <EBE6>
EB9C error? >>EB92
EB9E doesn't need checkpoint now
EBA3 exit

```

ProDOS MLI -- V1.1.1 -- 18 SEP 84      NEXT OBJECT ADDR: EBA3  
 ADDR    DESCRIPTION/CONTENTS

```

EBA4 ***** READ BITMAP *****
EBA4 save DEVNUM (FEA6)
EBA7 copy block offset wanted (FE91)
EBB1 BITMAP BLOCK = BITMAP PTR + BLOCK OFFSET (D91A)
EBBF set up read command

*** READ OR WRITE BITMAP ***

EBC1 save I/O command
EBC7 device = bitmap device (FEA6)
EBCD block = bitmap block (FEA7)
EBD7 point to bitmap buffer (EA9C)
EBDA do the I/O <EBF5>
EBDF restore old DEVNUM (BF30)
EBE2 ok? >>EBE5
EBE4 no, error exit
EBE5 RETURN

EBE6 ***** WRITE BITMAP *****
EBE6 set up write command
EBE8 and go do it >>EBC1

EBEA ***** WRITE BLOCK *****
EBEA set up write command
EBEC and go do it >>EBF0

EBEE ***** READ BLOCK *****
EBEE set up read command

EBF0 ***** READ OR WRITE BLOCK *****
EBF0 save I/O command
EBF2 where is my buffer? (E635)
EBF5 save flags
EBF6 and disable
EBF9 Set low byte of Buffer pointer
EBFB to zero
EBFD Initialize Global Page System error to 0 (BF0F)
EC00 set I/O transfer occurred flag
EC05 set unit to do I/O on (BF30)
EC0A do block I/O <DEDA>
EC0D error? >>EC12
EC0F no errors, restore things and exit
EC11 RETURN

```

ProDOS MLI -- V1.1.1.1 -- 18 SEP 84 NEXT OBJECT ADDR: EC11  
ADDR DESCRIPTION/CONTENTS

ProDOS MLI -- V1.1.1.1 -- 18 SEP 84 NEXT OBJECT ADDR: EC82  
ADDR DESCRIPTION/CONTENTS

EC12 error exit  
EC14 RETURN

EC15 \*\*\*\*\*  
\*\*\*\*\* MLI GET MARK CALL \*\*\*\*\*  
\*\*\*\*\*

EC15 copy mark to caller's list from FCB (FE92)  
EC25 exit with no errors  
EC26 RETURN

EC27 bad position error  
EC2A RETURN

EC2B \*\*\*\*\*  
\*\*\*\*\* MLI SET MARK CALL \*\*\*\*\*  
\*\*\*\*\*

EC2B set up to...  
EC33 copy user's mark to temporary  
EC35 new mark variable (FEA8)  
EC3A make sure it will not exceed EOF (D815)  
EC3F else, error >>EC27  
EC42 ---

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

EC48 get old mark (FE92)  
EC4B find its block no. (\*2) (D813)  
EC53 compute distance in pages from old mark's (FEAB)  
EC57 block to new mark (FE86)  
EC5D earlier -- need new data block >>EC6E  
EC61 too far forward -- need new block >>EC6E  
EC66 MSB's match? (D814)  
EC6B then mark is still in this block >>ED89  
EC6E check storage type (D807)  
EC71 zero? >>EC7A  
EC73 seedling, sapling or tree?  
EC77 no, special handling for DIR files >>EDBB

EC7A stomp on FCB2's mark?? (F300+\$52)  
EC7C (this should never happen anyway) (D800)  
EC7F and return with bad REFNUM error  
EC82 RETURN

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

EC83 copy storage type (D807)  
EC89 old data block needs writing? (D808)  
EC8E no >>EC95  
EC90 yes, do so <EE94>  
EC93 error? >>ECFE  
EC95 see if new mark is outside the range of (FE92)  
EC98 the current index block (D814)  
ECA7 yes >>ECC7  
ECAB yes >>ECC7  
ECAD no, same index block (FE96)  
ECB0 check storage type  
ECB1 sapling or tree are ok >>ED2D

\*\*\* SEEDLING \*\*\*

ECB3 seedling, check position (FEAB)  
ECB6 if position is outside of block 0..  
ECBA promote to sapling >>ED1B  
ECBC else, (D80C)  
ECC4 go get key block (seedling data block) >>ED7F

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

ECC7 does old index block need dumping? (D808)  
ECCC no >>ECD3  
ECCE yes, do so <EEA8>  
ECD1 error? >>ECFE  
ECD3 check storage type (FE96)  
ECD6 tree file?  
ECD8 yes >>ED00  
ECDA no, sapling (FEAC)  
ECDF is position in first index block?  
ECE2 no, need master index, subindex and data >>ED46  
ECE4 yes, first index, reset flags <EDAF>  
ECE7 is this a seedling?  
ECE8 if so, see if in first block >>ECB3

\*\*\* SAPLING \*\*\*

ECEA no, sapling, read its only index block <EE3B>  
ECED error? >>ECFE  
ECF2 set block no. of index block  
ECFC and continue below >>ED2D  
ECFE error exit  
ECFF RETURN

ProDOS MLI -- V1.1.1 -- 18 SEP 84      NEXT OBJECT ADDR: ECFE  
 ADDR    DESCRIPTION/CONTENTS

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

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

\*\*\* GET NEW INDEX BLOCK \*\*\*

```
ED1B need an index and data block
ED1D go allocate them >>ED46

ED20 set up block no. of subindex
ED28 read it <EELD>
ED2B error? >>ECFE
```

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

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

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

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

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

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

ProDOS MLI -- V1.1.1 -- 18 SEP 84      NEXT OBJECT ADDR: ED86  
 ADDR    DESCRIPTION/CONTENTS

\*\*\* GOT DATA BLOCK WANTED \*\*\*

```
ED89 ---
ED90 save previous mark in my variables (D812)
ED96 set new mark in the FCB (FEAA)
EDA1 ($4A/$4B --> data block buffer)
EDA3 $4C/$4D --> start of the page in
EDA5 the data block buffer which contains (FEAB)
EDA8 the mark.
EDAE exit
```

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

```
EDAF get flags (FE92)
EDB5 turn off low 3 bits (allocate no new
EDB7 blocks to file) (D808)
EDBA RETURN
```

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

```
EDBB DIR file?
EDBD yes! >>EDC4
EDBF no, bad storage type error
EDC1 go to SYSERR <BF09>
EDC4 else, get page distance (FE86)
EDC7 make it into blocks (divide by 2)
EDCE new position beyond old? (FEAB)
EDD1 yes >>EDE1
EDD3 else, use previous mark
EDD5 copy to BLKNUM <EDEF>
EDD8 error? >>EDFE
EDDA count it (FE9A)
EDDD more to skip? >>EDD3
EDDF no, got it >>ED89
EDE1 use next block pointer in DIR block
EDE3 copy to BLKNUM <EDEF>
EDE6 error? >>EDEE
EDE8 count it (FE9A)
EDEB more to skip >>EDE1
EDED got it now! >>ED89
```

\*\*\* COPY LINK TO BLKNUM \*\*\*

```
EDEF copy block number link
EDF1 to BLKNUM
EDF4 if non zero,
EDFA then go read block. >>EE00
EDFC else, EOF error
EDFE ---
EDFF RETURN
```

ProDOS MLI -- V1.1.1 -- 18 SEP 84      NEXT OBJECT ADDR: EDFF  
 -----  
 ADDR    DESCRIPTION/CONTENTS  
 -----

ProDOS MLI -- V1.1.1 -- 18 SEP 84      NEXT OBJECT ADDR: EE86  
 -----  
 ADDR    DESCRIPTION/CONTENTS  
 -----

```

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

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

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

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

*** I/O BLOCK ***
EE47 set I/O command
EE49 and block no. (D800)
EE53 must be non-zero block number
EE57 or horrible death!
EE5C fall through to read/write block (D801)

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

```

```

EE86 error? >>EE8B
EE88 no, exit normally
EE8A RETURN

EE8B else, exit with error
EE8D RETURN

EE8E ***** CHECKPOINT BITMAP & KEY BLOCK *****
EE8E checkpoint bitmap buffer <EB93>
EE91 go write key block for file >>EE37

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

EEA8 ***** CHECKPOINT INDEX BLOCK BUFFER *****
EEA8 checkpoint volume bitmap <EB93>
EEAB use $48/$49 buffer
EEAD block no. is current index block in FCB
EEB3 set to write
EEB5 go write it to disk <EE47>
EEB8 error? >>EEC5
EEBA no longer needs checkpoint
EEBC set flags accordingly (FE92)
EEC5 and exit

EEC6 ***** MLI OPEN CALL *****
*****

EEC6 search path for file <E5A3>
EEC9 found it? >>EECF
EECB no, bad path error
EECD exit >>EED6
EECF else, see if FCB already open on file <EFB3>
EED2 for write. if not, continue. >>EED8
EED4 else, file already open error
EED6 ---
EED7 RETURN

EED8 get FCB index (FE92)
EEDF free FCB found? >>EEE4
EEE0 no, all FCB's in use error
EEE3 RETURN

```



ProDOS MLI -- V1.1.1 -- 18 SEP 84      NEXT OBJECT ADDR: EEE3  
 ADDR    DESCRIPTION/CONTENTS

EEE4 zero out unused FCB  
 EEEF copy file ID fields to FCB  
 EEF2 (DEVNUM, DIR HDR BLK, DIR BLK, (FE92)  
 EEF5 DIR ENTRY NO.)  
 EF00 isolate storage type (FE5F)  
 EF08 and copy to FCB (D807)  
 EF0B get access (FE7D)  
 EF10 DIR file?  
 EF12 no >>EF16  
 EF14 yes, we are only reading (I hope)  
 EF16 update access flag in FCB (D809)  
 EF1B write protected? >>EF22  
 EF1D no, another FCB open on this file? (FE97)  
 EF20 yes, no touchie >>EED4  
 EF22 This line left over from version 1.0.1! (FE7C)  
 EF27 Now always jumps over error exit. >>EF2D  
 EF29 if bad, unsupported version error  
 EF2C RETURN

EF2D storage type must be < \$4  
 EF31 or equal to \$D  
 EF33 else, compatibility error >>EF29  
 EF35 ---  
 EF37 copy key block, blocks used, and  
 EF39 EOF mark to FCB (FE92)  
 EF49 BLKNUM = key block number  
 EF4E store REFNUM in FCB (FE9A)  
 EF54 go check and assign I/O buffer <FBED>  
 EF57 error? >>EF7D  
 EF59 go find VCB and set buff ptrs <E1EB>  
 EF5C set current level in FCB (BF94)  
 EF62 seedling, sapling or tree? (D807)  
 EF67 no, skip next stuff >>EF94  
 EF69 yes, make current mark in FCB outside  
 EF6B first index block to force a read of all (D814)  
 EF6E index blocks and BLOCK 0.  
 EF72 zero mark wanted, however (FEAA)  
 EF78 go set mark to zero <EC48>  
 EF7B ok? >>EF99  
 EF7D no, save the error code  
 EF81 got and I/O buffer? (D80B)  
 EF84 no >>EF8C  
 EF86 yes, free it <FC4A>  
 EF8C mark FCB not in use  
 EF92 exit with error  
 EF93 RETURN

ProDOS MLI -- V1.1.1 -- 18 SEP 84      NEXT OBJECT ADDR: EF93  
 ADDR    DESCRIPTION/CONTENTS

EF94 else, read key block to I/O buffer <EE04>  
 EF97 error? >>EF7D  
 EF99 bump open file count in VCB (FE91)  
 EF9F indicate files are open in VCB (D911)  
 EFA7 put REF NUM in caller's parmlist (FE92)  
 EFB1 exit with no errors  
 EFB2 RETURN

EFB3 \*\*\*\*\* FIND A FCB \*\*\*\*\*

EFB3 clear flags and index byte  
 EFB6 ---  
 EFBF found a free FCB yet? (FE93)  
 EFC2 yes >>EFC7  
 EFC4 no, bump entry count (FE9A)  
 EFC7 FCB in use? (D800)  
 EFCA yes >>EFD9  
 EFCC no,  
 EFCF save index to free FCB (FE92)  
 EFD2 flag that we found one  
 EFD7 and skip this FCB >>EFF7  
 EFD9 ---  
 EFD9 compare file ID's to see if this FCB (D8J0)  
 EFE2 is open on the requested file. (FE58)  
 EFE5 no match? >>EFF7  
 EFEB indicate FCB already open on file (FE97)  
 EFEE write enabled? (D809)  
 EFF3 if not, allow multiple open access to file >>EFF7  
 EFF5 else, error exit  
 EFF6 RETURN

EFF7 return index to start of FCB  
 EFFB bump to next FCB  
 EFFD and loop >>EFBE  
 EFFF when done, exit normally  
 F000 RETURN

F001 \*\*\*\*\* MLI READ CALL \*\*\*\*\*  
 \*\*\*\*\* MLI READ CALL \*\*\*\*\*

F001 point to data buffer <F20D>  
 F004 copy request length <F1F2>  
 F007 save access  
 F008 set up marks <F21F>  
 F00C read access permitted?  
 F00E yes >>F014  
 F010 no, access error  
 F014 will we read past EOF? >>F03B  
 F016 yes, (FE92)

PRODUS MLI -- V1.1.1 -- 18 SEP 84 NEXT OBJECT ADDR: F019

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

F019 LENGTH = EOF - current mark (D815)  
 F031 are we already at EOF? (FEDA)  
 F034 no >>F046  
 F036 Yes, EOF error  
 F03B else, zero length request? (FEDA)  
 F041 no >>F046  
 F043 Yes, set mark and exit >>F0F9  
 F046 validity check data buffer <FC82>  
 F049 no good? >>F038  
 F04B ok, get storage type for file <F218>  
 F04E standard kind of file?  
 F050 Yes >>F055  
 F052 NO, DIR file >>F1B8  
 F055 else, set mark (to read proper buffers) <EC48>  
 F058 error? >>F038  
 F05A set up buffer indexing <F110>  
 F05D move all that can be moved out of data buff <F13A>  
 F060 newline or len=0: exit now! >>F043  
 F062 newline enabled? continue block by block >>F055  
 F064 at least 1 block's worth left to be read? (FEAE)  
 F068 if not, never mind >>F055  
 F06A if so, store block count wanted (FEAF)  
 F06D get FCB flags <F606>  
 F070 data block modified?  
 F072 Yes, continue block by block for now >>F055

\*\*\* FAST DIRECT READ ROUTINE \*\*\*

F074 signal no read occurred yet (FEB2)  
 F077 read directly into caller's data buffer  
 F07F set mark/read data block to caller's buff <EC48>  
 F082 error? >>F0ED  
 F084 bump buffer pointer to next location  
 F088 drop length remaining by 512 bytes (FEAE)  
 F08E bump mark (FEAB)  
 F096 and mark's MSB as necessary (FEAC)  
 F099 check if we are out of index block (FEAC)  
 F09F drop counter of multi-blocks (FEAF)  
 F0A2 and keep on >>F0B1  
 F0A4 end of multi-block read, put ptrs back <F1AD>  
 F0A7 more to read? (FEAD)  
 F0AD no, exit through finish-up >>F0F9  
 F0AF Yes, conventional block by block read then >>F055

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

NEXT OBJECT ADDR: F0AF

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

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

\*\*\* ERROR CLEANUP \*\*\*

----  
 F0FC  
 F0ED  
 F0EE set buffer ptrs/VCB <F1AD>  
 F0F2  
 F0F3 finish up I/O <F0F9>  
 F0F7 exit with error  
 F0F8 RETURN

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

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

F110 \*\*\*\*\* SET UP BUFFER INDEXING \*\*\*\*\*

----  
 F110  
 F114 back up pointer to data buffer by an  
 F116 amount equal to the LSB of the mark (FEAA)  
 F119 (which makes indexing easier)  
 F11F newline mode enabled? (D81F)  
 F123 no, CLC >>F12F  
 F125 Yes, SEC  
 F126 copy newline mask (FEB1)  
 F129 and newline character (D80A)  
 F12F first char index is LSB of mark in YREG (FEAA)  
 F132 \$4C/\$4D --> page containing mark  
 F136 request count LSB in XREG (FEAD)  
 F139 exit

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

```

F13A ***** 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

---
F13A partial page to move? >>F145
F13B no, any full pages left? (FEAE)
F140 no, read complete >>F194
F142 yes, drop MSB of request length (FEAE)
F145 ---
F146 copy one byte $4C ---> $4E
F14B end of requested chunk? >>F168
F14D no, newline enabled? >>F17D
F14F ---
F151 no, loop for more >>F146
F153 end of page, bump pointers
F157 bump new mark (FEAB)
F15F finished first page of block buffer?
F163 if so, continue >>F146
F166 no, need another block from disk >>F197
F168 another page in request length? (FEAE)
F16B no >>F187
F16E more in this block-page? >>F176
F170 no, on last page of block?
F174 no >>F179
F176 yes, drop request len by one page (FEAE)
F179 back up to next byte again
F17A go copy next page >>F14D

F17D check for newline
F185 not it, never mind! >>F14F
F187 else, were we done with page?
F188 no >>F194
F18A yes, bump pointer
F18C and mark (FEAB)
F194 set overflow flag (read completed) (FIAC)

F197 update mark LSB (FEAA)
F19C bump request count if necessary
F19D update count LSB (FEAD)
F1A3 point beyond data in caller's buffer
F1AB ---
FIAC and exit

```

ProDOS MLI -- V1.1.1 -- 18 SEP 84      NEXT OBJECT ADDR: FIAC  
 ADDR    DESCRIPTION/CONTENTS

```

FIAD ***** CLEANUP AFTER DIRECT I/O *****
FIAD restore caller's data buffer pointer
FI8B go set buffers/find VCB and exit >>E1EB

FI8B ***** DIRECTORY FILE READ *****
FI8B set mark/read <EC48>
FI8E error? >>F1EF
FI8F set up buffer indexing <F110>
FI93 move data from I/O buffer <F13A>
FI96 need next block? >>F18B
FI98 no, finish up I/O <F0F9>
FI9B ok? exit >>F1ED
FI9D not ok. EOF error?
FI9E no, out now >>F1EE
FI9F yes, point beyond EOF anyway? <ED89>
F1D5 zero out data block I/O buffer <ED67>
F1DD dummy up an empty DIR block with previous (D810)
F1E0 pointer and no forward pointer in I/O
F1E2 buffer.
F1E4 zero out current block no. (D810)
F1ED return to caller
F1EE RETURN

F1EF finish up and error exit >>F0F2

FI2 ***** COPY CALLER'S I/O LENGTH *****
FI2F copy request length to LENGTH and
FI4 a temporary variable
FI5 pick up ACCESS flcgs for file (FE92)
FI8B exit to caller
FI9C RETURN

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

F20D set up pointer
F218 YREG --> FCB (FE92)
F21B AREG = storage type (D807)
F21E exit

F21F ***** COPY FILE MARK AND COMPUTE *****
***** AND COMPARE END MARK *****

```

PRODOS MLI -- V1.1.1 -- 18 SEP 84      NEXT OBJECT ADDR: F21F  
 ADDR    DESCRIPTION/CONTENTS

F21F    ---  
 F225    copy file mark (D812)  
 F22B    and set previous mark also (FE8D)  
 F22E    add length giving new mark in scratch area (FEDA)  
 F235    (3 byte addition)  
 F23D    will new mark exceed EOF? (FE86)  
 F24B    return with carry set accordingly

F24C    \*\*\*\*\* SET NEW MARK & EOF \*\*\*\*\*  
 F24C    set up indexes <F27E>  
 F24F    set new EOF in FCB (FE8A)  
 F255    and new mark (FE8D)  
 F25B    save new mark in scratch variable too (FE86)  
 F262    does mark exceed EOF? <F27E>  
 F265    if so, we must extend EOF <F23D>  
 F26B    save old EOF (D815)  
 F273    set new EOF to mark if necessary (FE86)  
 F279    ---  
 F27D    exit

F27E    subroutine to set 3 byte indexes  
 F285    RETURN

F286    \*\*\*\*\* MLI WRITE CALL \*\*\*\*\*  
 \*\*\*\*\*

F286    copy request length <F1F2>  
 F28A    copy file mark <F21F>  
 F28D    extend EOF if needed <F268>  
 F291    write access enabled?  
 F293    yes >>F299  
 F295    no, access error  
 F299    check status of this device <F458>  
 F29C    error? >>F2D9  
 F29E    request length = 0? (FEDA)  
 F2A4    no >>F2A9  
 F2A6    yes, exit through finish-up >>F0F9

F2A9    find caller's data buffer <F20D>  
 F2AC    check storage type  
 F2AE    if DIR file, error >>F295  
 F2B0    set mark/read blocks <EC48>  
 F2B3    error? >>F2D9  
 F2B5    get FCB flags <F606>  
 F2B8    any new blocks needed?  
 F2BA    no >>F31E  
 F2BC    yes, allocating them  
 F2BE    ---

PRODOS MLI -- V1.1.1 -- 18 SEP 84      NEXT OBJECT ADDR: F2BF  
 ADDR    DESCRIPTION/CONTENTS

F2BF    count number of blocks needed  
 F2C2    store number needed (FE94)  
 F2C8    see if the blocks are available <E973>  
 F2CB    no, disk full >>F2D9  
 F2CD    yes, get FCB flags <F606>  
 F2D0    master index block needed?  
 F2D2    no >>F2E1  
 F2D4    yes, go add it <F399>  
 F2D7    and go on if no errors >>F2ED

F2D9    error,  
 F2DA    set new mark/EOF <F24C>  
 F2DE    and finish I/O, exit with error >>F0F2

F2E1    check FCB flags again <F606>  
 F2E4    need sub-index block?  
 F2E6    no >>F2ED  
 F2E8    yes, go do it <F3E4>  
 F2EB    error? >>F2D9  
 F2ED    buy a new block for data <F438>  
 F2F0    error? >>F2D9  
 F2F2    get FCB flags <F606>  
 F2F5    indicate index buffer changed  
 F2F7    no new blocks needed now  
 F2F9    update FCB flags (D808)  
 F2FF    make index block offset from mark  
 F307    store new block no. in index block (FE87)  
 F314    and store it as current data block (FE92)  
 F31E    set up buffer indexing <F110>  
 F321    start writing <F329>  
 F324    go see if more blocks are needed >>F2B0  
 F326    I/O finish up when done >>F0F9

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

F329    ---  
 F32C    lower request count by 1 (FEAE)  
 F334    ---  
 F335    copy partial page from caller's data  
 F337    to I/O block buffer  
 F33C    ---  
 F33F    next page in caller's area  
 F343    bump mark by \$100 (FEAB)  
 F34B    still in same I/O block page?  
 F34F    yes >>F334  
 F352    no, clear overflow (I/O incomplete) >>F379

ProDOS MLI -- V1.1.1 -- 18 SEP 84      NEXT OBJECT ADDR: F352  
 ADDR    DESCRIPTION/CONTENTS

F354 any complete pages left to write? (FEAE)  
 F357 no >>F369  
 F359 yes, more in this page?  
 F35A yes >>F362  
 F35C no, first block-page?  
 F360 no >>F365  
 F362 yes, one less complete page to do (FEAE)  
 F365 readjust index  
 F366 continue with full page >>F33C

---  
 F369 a few bytes left to write? >>F376  
 F36A no, bump data buffer by \$100  
 F36C and mark (FEAB)  
 F36E  
 F376 set overflow (I/O complete) (FLAC)  
 F379 store LSB of mark (FEAA)  
 F37C and of request count (FEAD)  
 F380 indicate data block modified <F606>  
 F383 and DIR entry needs update  
 F389 advance pointer into caller's buffer (FEAA)  
 F394 set FCB flag to indicate write occurred <FA66>  
 F398 exit

F399 \*\*\*\*\* ADD NEW MASTER INDEX BLOCK \*\*\*\*\*  
 (MAKE A TREE FILE)

F399 add higher level <F3F1>  
 F39C error? >>F3F0  
 F39E get storage type <F218>  
 F3A1 tree?  
 F3A3 yes >>F3AA  
 F3A5 no, add another level <F3F1>  
 F3A8 error? >>F3F0  
 F3AA buy another block <F438>  
 F3AD error? >>F3F0  
 F3AF male offset into current index block (FEAC)  
 F3B2 from current mark  
 F3B4 point index to new block (FE86)  
 F3C3 also save as current data block (FE92)  
 F3CD checkpoint bitmap & key block <EE8E>  
 F3D0 error? >>F3F0  
 F3D5 zero out new index block  
 F3DC ---  
 F3E3 and exit

ProDOS MLI -- V1.1.1 -- 18 SEP 84      NEXT OBJECT ADDR: F3E4  
 ADDR    DESCRIPTION/CONTENTS

F3E4 \*\*\*\*\* ADD NEW INDEX BLOCK \*\*\*\*\*  
 F3E4 check storage type <F218>  
 F3E9 seedling? >>F3F1  
 F3EB no, read key index block <EE3B>  
 F3EE and go add data block >>F3AA  
 F3F0 exit if error occurs

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

F3F1 buy a block <F438>  
 F3F4 error? >>F437  
 F3F9 save old key block number (D80C)  
 F401 make new block the key block (D80C)  
 F40E and current index block in FCB (D80F)  
 F417 store pointer to old key block  
 F41A in first position of new index  
 F421 checkpoint bitmap and new key block <EE8E>  
 F424 error? >>F437  
 F426 get storage type <F218>  
 F42B upgrade it to next higher type (D807)  
 F42E indicate DIR entry needs update (D808)  
 F437 exit

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

F438 allocate a disk block <EAB6>  
 F43B error? >>F457  
 F43D get FCB flags <F606>  
 F440 indicate DIR entry needs update  
 F449 add 1 to blocks in use for file  
 F456 ---  
 F457 exit

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

F458 get FCB flags <F606>  
 F45B any buffers in use? (I/O activity)  
 F45D if so, assume its ok >>F456  
 F45F no, (D801)  
 F462 select new device (BF30)

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

F465 Save Unit Number  
 F467 Save Block Number on stack  
 F46D Indicate Status call  
 F471 Indicate Block 0  
 F475 Go do I/O <DEDA>  
 F478 Restore Block Number to original value  
 F480 Exit

```

ProDOS MLI -- V1.1.1 -- 18 SEP 84      NEXT OBJECT ADDR: F481
-----
ADDR  DESCRIPTION/CONTENTS
-----

```

```

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

```

```

F481 check REF NUM
F485 specific close? >>F4BC

```

```

*** CLOSE ALL OPEN FILES ***

```

```

F487 no errors yet (FEBE)
F48C store FCB index (FE92)
F490 get its level (D81B)
F493 if below system LEVEL, skip it (BF94)
F496 yes, skip it >>F4AD
F498 no, active FCB? (D800)
F49B no >>F4AD
F49D yes, flush it and update directory <F51E>
F4A0 error? >>F4EF
F4A2 no, close specific FCB <F4C1>
F4A7 is this a close-all?
F4A9 yes, ignore errors >>F4AD
F4AB no, stop on error >>F4EF
F4AD bump FCB index to next one (FE92)
F4B3 and continue >>F48C
F4B5 when done, load error number (FEBE)
F4BB and exit

```

```

*** CLOSE SPECIFIC FILE ***

```

```

F4BC flush it <F526>
F4BF error? >>F4EF
F4C1 get buffer number (FE92)
F4C7 free its pages <FC4A>
F4CA error? >>F4EF
F4CC release FCB
F4D4 set DEVNUM (D801)
F4DA find VCB for device <E876>
F4DD decrement count of open files in VCB (FE91)
F4E3 some are open... >>F4ED
F4E5 if all are closed, turn off (D911)
F4E8 "files open" flag
F4ED ---
F4EE exit
F4EF jump to handle close error >>F5F7

```

```

ProDOS MLI -- V1.1.1 -- 18 SEP 84      NEXT OBJECT ADDR: F4EF
-----
ADDR  DESCRIPTION/CONTENTS
-----

```

```

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

```

```

F4F2 flush specific file?
F4F6 yes >>F526
F4F8 no, clear flush-all error code (FEBE)
F4FB do all FCBs
F4FD set FCB index for next FCB (FE92)
F501 is this file open? (D800)
F504 no >>F50B
F506 yes, flush it <F51E>
F509 error? >>F51B
F50B bump to next FCB (FE92)
F511 and go flush it too >>F4FD
F513 ---
F514 return with error code if any (FEBE)
F51A RETURN

```

```

F51B ---

```

```

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

```

```

F51E find buffer/VCB <E1EB>
F521 no error? >>F530
F523 error - exit >>F5F7

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

```

ProDOS MLI -- V1.1.1 -- 18 SEP 84 NEXT OBJECT ADDR: F56F

ProDOS MLI -- V1.1.1 -- 18 SEP 84 NEXT OBJECT ADDR: F610

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

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

F56F BLKNUM = current DIR block (FE5A)  
 F579 read DIR block <EBEE>  
 F57C error? >>F51B  
 F57E copy directory header <E6AB>  
 F581 are we in block with this file's entry? (FE5C)  
 F58A no >>F591  
 F58F yes >>F598  
 F591 no, set new block number  
 F595 read it <EBEE>  
 F598 point at directory entry in block <E49D>  
 F59B copy file entry from directory <E5A8>  
 F5A1 copy blocks used count to entry (D818)  
 F5AF copy new EOF (D815)  
 F5BA and new key block no. (D80C)  
 F5C3 isolate new storage type (D805)  
 F5CD combine it with name length (FE5F)  
 F5D5 and update type/len field in entry (FE5F)  
 F5D8 write entry back to directory <E4B6>  
 F5DB error? >>F5F7  
 F5E0 turn off "write occurred" flag (D81C)  
 F5E8 same bitmap in memory (FE59)  
 F5EE no, exit now >>F5F5  
 F5F0 yes, checkpoint it also <EB93>  
 F5F5 no errors, exit  
 F5F6 RETURN

F611 \*\*\*\*\* MLI SET EOF CALL \*\*\*\*\*  
 \*\*\*\*\*  
 get storage type <F218>  
 F614 if DIR file...  
 F616 its an access error >>F60D  
 F618 else, save type for truncate to  
 F619 mess with.  
 F61F write access permitted? (D809)  
 F624 no, error >>F60D  
 F626 check device status <F458>  
 F629 error? >>F60D  
 F632 copy EOF from FCB (D815)  
 F640 copy caller's new EOF  
 F64B compare old EOF to new (FE8A)  
 F651 if less than or equal to... >>F658  
 F653 if greater... >>F66D  
 \*\*\* OLD EOF <= NEW EOF \*\*\*  
 \*\*\* NO TRUNCATE NEEDED \*\*\*  
 F658 new eof beyond old  
 F65F copy caller's EOF to FCB  
 F66A exit by indicating flush needed >>FA66

F5F7 \*\*\*\*\* CLOSE ERROR \*\*\*\*\*  
 F5F7 is this a close or flush all?  
 F5FC no >>F604  
 F600 yes, save error code (FEBE)  
 F603 RETURN  
 F604 else, real error right now  
 F605 RETURN  
 F606 \*\*\*\*\* GET FCB FLAGS \*\*\*\*\*  
 F606 load FCB flags (FE92)  
 F609 from FCB (D808)  
 F60C and exit  
 F60D \*\*\*\*\* FILE ACCESS ERROR \*\*\*\*\*  
 F60D exit with file access error code  
 F610 RETURN

\*\*\* OLD EOF > NEW EOF \*\*\*  
 (\*\* TRUNCATE FILE  
 flush first <F526>  
 F670 error? >>F610  
 F672 \$43/\$49 --> end of data block I/O buffer  
 F67C compare current mark to new EOF (FE92)  
 F689 it is prior to EOF >>F6A2  
 F691 if past EOF, force mark back to EOF (FE92)  
 F6A2 construct EOF block number and (FEAA)  
 F6A5 byte offset into block from new (FEC6)  
 F6A8 EOF mark. (FEAB)  
 F6C0 on a block boundary? (FEC7)  
 F6C3 yes >>F6E2  
 F6C5 no, (FEC5)  
 F6C9 decrement block by 1  
 F6D7 but don't let it fall below 0  
 F6E2 copy key block number (FE92)  
 F6F1 set blocks freed to zero  
 F6F9 truncate file at new EOF <FA78>  
 F6FC save status  
 F704 set new key block in FCB (FEBF)  
 F70A drop FCB block count by number (D818)  
 F70D of blocks freed in truncate routine. (FEC2)

ProDOS MLI -- V1.1.1 -- 18 SEP 84      NEXT OBJECT ADDR: F71A  
 -----  
 ADDR    DESCRIPTION/CONTENTS  
 -----

F71A copy new storage type (FECl)  
 F727 turn off all block allocation flags <EDAF>  
 F72A update VCB free block count <F9F3>  
 F73A copy mark (D812)  
 F73C force current mark to infinity (D812)  
 F743 go set mark <EC48>  
 F746 no errors? >>F74F  
 F748 if error, indicate in saved status  
 F74E but continue  
 F74F copy caller's EOF to FCB <F658>  
 F752 Flush and update <F526>  
 F755 no errors? >>F75E  
 F757 if error, indicate in saved status  
 F75D but continue  
 F75E ---  
 F760 exit

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

---  
 F761 copy EOF to caller's list (D815)  
 F772 exit -- no errors

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

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

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

F785 get the file entry <E5A3>  
 F788 ok? >>F7CC  
 F78A no, bad path?  
 F78D no, real error >>F7E9  
 F78F else, make it VOL DIR type  
 F791 with name length = 0 (FE5F)  
 F796 no free blocks needed (FE94)  
 F79C go through the motions to update the (FE91)  
 F79F VCB block count. <E97E>  
 F7A5 copy blocks free from VCB (D915)  
 F7B1 copy total blocks on volume to AUX\_ID (D913)  
 F7BF total - free = blocks used (FE94)  
 F7CC shift type down from high nibble (FE5F)

ProDOS MLI -- V1.1.1 -- 18 SEP 84      NEXT OBJECT ADDR: F7D8  
 -----  
 ADDR    DESCRIPTION/CONTENTS  
 -----

F7D8 copy the data to caller's parmlist (FE0C)  
 F7E9 and exit

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

F7EA get the file entry <E5A3>  
 F7ED error? >>F814  
 F7EF indicate backup needed now (BF95)  
 F7FE copy 13 parms from caller's list to (FE0C)  
 F801 file entry staging area >>F808  
 F808 ---  
 F80D if any spurious access bits are on...  
 F811 access error!  
 F814 RETURN

F815 else, anything in his modification date?  
 F819 no >>F81E  
 F81B yes, go update directory >>E4C6

F81E no, use system date then update directory >>E4B6

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

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

\*\*\* RENAME VOLUME \*\*\*

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

F844 make type/len for a VOL DIR HDR  
 F84B write new name to VOL HDR <F93C>  
 F84E error? >>F8B9  
 F855 copy new name to device's VCB (D700)  
 F861 exit, no errors  
 F862 RETURN



ProDOS MLI -- V1.1.1 -- 18 SEP 84      NEXT OBJECT ADDR: F862  
 ADDR    DESCRIPTION/CONTENTS

\*\*\* RENAME FILE \*\*\*

F863 get path index <F959>  
 F866 copy old name with prefix to my buffer (D700)  
 F872 copy new name to buffer <F94B>  
 F875 error? >>F8B9  
 F877 get path index <F959>  
 F87D compare all levels of names up to and (DC00)  
 including the last. Find first which  
 differ.  
 F881 save indices into names which point to (FEB9)  
 F885 final name. (FBA)  
 F888 ---  
 F895 exit if they match completely  
 F896 RETURN

F897 index to differing new name (FEB9)  
 F89A point past it (D700)  
 F8A2 must be the last! (D700)  
 F8A5 it isn't >>F8B7  
 F8A7 it is, (FBA)  
 F8AA do the same with the old name (DC00)  
 F8B5 difference is only in last index? >>F8BB  
 F8B7 no, bad path error  
 F8B9 ---  
 F8BA RETURN

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

F8C4 if error, better be file not found  
 F8C6 or else its really an error... >>F8B9  
 F8C8 copy old pathname again <E08A>  
 F8CB get its file entry <E5A3>  
 F8CE error? >>F8B9  
 F8D0 search FCB's <EFB3>  
 F8D5 exit if the file is open for write >>F8B9  
 F8DA does ACCESS permit rename?  
 F8DE no, access error  
 F8E0 ---  
 F8E1 RETURN

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

ProDOS MLI -- V1.1.1 -- 18 SEP 84      NEXT OBJECT ADDR: F8EF  
 ADDR    DESCRIPTION/CONTENTS

F8EF else, compatibility error  
 F8F3 copy new path again <F94B>  
 F8F6 error? >>F8B9  
 F8F8 get length of last name (FEB9)  
 F903 copy it and name to file entry buffer (D700)  
 F913 combine new len with type (D700)  
 F919 DIR file?  
 F91B no, go update entry and exit >>F939  
 F91D yes, (FE70)  
 F927 read key block of this subdirectory <EBEE>  
 F92A error? >>F8B9  
 F92F copy new name to DIR HDR (D700)  
 F934 and update directory's key block <F93C>  
 F937 error? >>F8B9  
 F939 go update directory entry and exit >>E4C6

F93C \*\*\*\*\* COPY PATH TO BUFF & WRITE \*\*\*\*\*

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

F94B \*\*\*\*\* POINT TO NEW NAME \*\*\*\*\*  
 COPY TO BUFFER

F94B \$48/\$49 --> second pathname/  
 F956 go copy it >>E095

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

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

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

F966 get file entry <E5A3>  
 F969 error? >>F9B5  
 F96B find FCB if any <EFB3>  
 F96E FCB open? (FE97)  
 F971 no >>F977  
 F973 yes, file open error  
 F976 RETURN

F977 no free blocks needed  
 F97F go compute VCB free block count <E973>  
 F982 ok? >>F989  
 F984 error, disk full?  
 F987 no, real error >>F9B5

```

PRODOS MLI -- V1.1.1 -- 18 SEP 84          NEXT OBJECT ADDR: F989
-----
ADDR  DESCRIPTION/CONTENTS
-----

```

```

F989 DESTROY enabled in ACCESS? (FE7D)
F98E yes >>F995
F990 no, access error
F995 check status of device (BF34)
F99B error? >>F9B5
F99D point to key block (FE70)
F9AC DIR file?
F9B0 no >>F9B6
F9B2 yes, handle differently >>FA0E
F9B5 RETURN

```

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

```

F9B6 set new storage type (FECl)
F9BD zero EOF mark (FECl)
F9C3 byte offset = $200
F9C8 free all blocks in file <FA78>
F9CB error? >>F9B5
F9CD free key block of seedling (FEC0)
F9D6 error? >>F9B5
F9D8 mark DIR entry free
F9DD decrement DIR file count (FE53)
F9E8 checkpoint volume bit map <EB93>
F9EB error? >>F9B5
F9ED update free block count in VCB <F9F3>
F9F0 and go update the directory >>E4B6

```

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

```

F9F3 add blocks freed to total free blocks (FE91)
F9F6 in VCB. (FEC2)
FA08 start next search for free blocks at
FA0A start of bitmap. (D91C)
FA0D exit

```

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

```

FA0E DIR file?
FA10 no, error >>FA61
FA12 read volume bitmap block <EB64>
FA15 error? >>FA60
FA17 BLKNUM = key block pointer (FE70)
FA21 read it <EBEE>
FA24 errors? >>FA60
FA26 if DIR has any files... (DC25)
FA30 access error
FA35 write back block marking entry free (DC44)
FA3B error? >>FA60
FA3D if "next_pointer" is zero.... (DC02)

```

```

PRODOS MLI -- V1.1.1 -- 18 SEP 84          NEXT OBJECT ADDR: FA47
-----
ADDR  DESCRIPTION/CONTENTS
-----

```

```

FA47 go back and pretend it's a seedling >>F9CD
FA49 else, (DC03)
FA4C free next block <EA34>
FA4F error? >>FA60
FA51 BLKNUM = next block (DC02)
FA5B read it <EBEE>
FA5E if ok, continue in loop >>FA3D
FA60 else, error exit

```

FA61 incompatible file format error

FA66 \*\*\*\*\* SET WRITE OCCURRED FLAG \*\*\*\*\*

```

FA66 save some registers
FA69 indicate write occurred (FE92)
FA74 restore registers and exit
FA77 RETURN

```

FA78 \*\*\*\*\* TRUNCATE FILE AT EOF \*\*\*\*\*

```

FA78 check storage type*16 (FECl)
FA7B seedling?
FA7D yes >>FA8A
FA7F no, sapling?
FA81 yes >>FA8D
FA83 no, tree?
FA85 yes >>FA90
FA87 no, die horribly <BF0C>
FA8A go to seedling truncate >>FB5C
FA8D go to sapling truncate >>FB23

```

```

FA90 truncate tree,
FA92 at most 128 blocks in master index (FEC8)
FA95 read the master index <FB87>
FA98 error? >>FAF5
FA9A at EOF yet? (FEC8)
FAA0 yes >>FAF6

```

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

```

FAA2 copy up to 8 non-zero index block
FAA4 numbers to (DC00)
FAA7 a handy table (FECA)
FAB8 ---
FAC1 if there weren't 8 left to do, zero (FECA)
FAC4 remainder of the table (FED2)
FACA ---

```

Beneath Apple ProDOS Supplement

PRODOS MLI -- VI.1.1 -- 18 SEP 84 NEXT OBJECT ADDR: FB51

PRODOS MLI -- VI.1.1 -- 18 SEP 84 NEXT OBJECT ADDR: FACB

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

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

FB52 back to block 0? (FEC4)  
FB55 no >>FB3D  
FB57 Yes, demote to seedling <FB94>  
FB5A error? >>FB86

FACB update master index counter (FEC8)  
FAD0 for all 8 entries: (FEC9)  
FAD3 set BLKNUM (FECA)  
FADB (exit when a 0 entry is found) >>FA95  
FAE2 read the sub-index block <EBEE>  
FAE5 error? >>FAF5  
FAE7 free all its blocks <FB86>  
FAEA error? >>FAF5  
FAF0 and loop to do all 8 >>FAD0  
FAF2 then go back and reread master index >>FA95  
FAF4 normal exit  
FAF5 RETURN

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

FB5C read key block <FB87>  
FB5F error? >>FB86  
FB61 first page? (FEC7)  
FB64 Yes >>FB6C  
FB67 no, better be second >>FB85  
FB69 get byte offset (FEC6)  
FB6C ---  
FB6E zero beyond EOF mark (DD00)  
FB7C in both pages if necessary (DC00)  
FB82 then write block back and exit >>EBEA

FAF6 now go free all the sub-index blocks (FEC4)  
FAFA which follow EOF <FB88>  
FAFD error? >>FAF5  
FAFF write back master index <EBEA>  
FB02 error? >>FAF5  
FB04 EOF in first subindex? (FEC4)  
FB07 if so, demote to sapling fkle >>FB1E  
FB09 else, BLKNUM = subindex block which (DC00)  
FB0C contains the EOF mark  
FB11 (exit if none there) >>FAF4  
FB18 else, read subindex block <EBEE>  
FB1B and continue below >>FB28  
FB1D unless there is an error

FB85 exit normally  
FB86 RETURN

FB23 read key block <FB87>  
FB26 error? >>FAF5  
FB28 get LSB of block number (FEC5)  
FB2C if zero, no blocks to free >>FB38  
FB2E else, free rest of blocks in index <FB88>  
FB31 following the EOF. check for error >>FAF5  
FB33 write index block back <EBEA>  
FB36 error? >>FAF5  
FB38 get LSB of block number (FEC5)  
FB3B might be block 0? >>FB52  
FB3D no, get BLKNUM of data block (DC00)  
FB40 from index block  
FB45 (no block allocated?) >>FAF4  
FB4C read data block <EBEE>  
FB4F and continue below >>FB61  
FB51 unless error occurred

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

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

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

FB94 free block (FEC0)  
FB9D error? >>FB85  
FB9F get block from old index (DC00)  
FBAC reduce storage type by one (FEC1)  
FBB4 and exit  
FBB5 RETURN

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

--- save BLKNUM  
FB88 for each index entry after mark, (FE9D)  
FB89 if it is non-zero....  
FB90 free the block <EA34>  
FB93 error? >>FB84  
FB95 zero the index entry now (FE9D)  
--- loop through all entries >>FB8E  
--- restore old BLKNUM  
FB86 and exit

ProDOS MLI -- V1.1.1 -- 18 SEP 84      NEXT OBJECT ADDR: FBEC  
 ADDR    DESCRIPTION/CONTENTS

```

FBED ***** ALLOCATE I/O BUFFER *****
  ---
FBEF get I/O buffer page number
FBF2 can't be below $800
FBF4 else, error >>FC38
FBF6 can't be above $BC00
FBF8 else, error >>FC38
FBFD $4A/$4B --> I/O buffer
FC01 must be page aligned! >>FC38
FC07 ---
FC08 check each page of I/O buffer for <FC73>
FC0B prior allocation in system bit map (BF58)
FC18 ---
FC19 if ok, mark each page as allocated <FC73>
FC1C in system memory bit map (BF58)
FC29 assign buffer number (REFNUM*2) in FCB (D800)
FC31 and save buffer location in buffer list
FC36 exit
FC37 RETURN

FC38 bad I/O buffer error
FC3B RETURN

FC3C ***** LOCATE I/O BUFFER *****
  ---
FC3D AREG contains buffer number *2 (BF6E)
FC40 move buffer pointer to NXTBUF variable (FEDD)
FC49 exit

FC4A ***** FREE I/O BUFFER *****
FC4A is buffer already free? <FC3C>
FC4F yes, exit >>FC71
FC53 zero its address in system global page (BF6F)
FC60 ---
FC61 free each page in buffer <FC73>
FC64 by marking system bit map
FC71 exit
FC72 RETURN

FC73 ***** LOCATE BIT MAP POSITION *****
      (GIVEN PAGE NUMBER)
FC73 XREG contains page number
FC74 compute page number times 8
FC77 use as offset for bitmask (FE00)
FC7E page number / 8 = byte offset
FC7F into bitmap
  
```

ProDOS MLI -- V1.1.1 -- 18 SEP 84      NEXT OBJECT ADDR: FC81  
 ADDR    DESCRIPTION/CONTENTS

```

FC81 exit

FC82 ***** CHECK BUFFER VALIDITY *****
      START > $200    END < $BF00
FC82 get buffer address (MSB)
FC86 must be >$200 else error >>FC38
FC88 get length (FEDB)
FC8E compute last page no. of buffer
FC93 ---
FC9A may not extend into $BF00
FC9C else, error >>FC38

      *** CHECK IF BLOCK OF MEMORY IS FREE ***
  ---
FC9F see if this page is allocated <FC73>
FA6 if so, error >>FC38
FA8 else, check other pae also
FCAC then exit if both have been checked
FCAD RETURN

FAEA ***** MLI GET BUFF CALL *****
      ***** MLI SET_BUFF CALL *****
FAEA get next available buffer
FCB3 put its address in caller's parmlist
FCBB and exit
FCBC RETURN

FCBD ***** MLI SET_BUFF CALL *****
      ***** MLI SET_BUFF CALL *****
FCBD mark his buffer allocated
FCC2 error? >>FCE4
FCC4 get old buffer address (FEDE)
FCCE free old buffer's pages in map <FC59>
FCD5 copy old buffer contents
FCD7 to new buffer
FCE3 then exit
FCE4 RETURN

FCE5 ***** GO TO QUIT CODE HANDLER *****
FC83 enable 2nd 4K bank of language card (C083)
FC88 (it lives at $D100-$D3FF) (C083)
FC8B Save zeropage $00 through $03 on stack
FCF7 Set ($00) -> $D100
FCF9 Set ($02) -> $1000
  
```

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

PRODOS MLI -- V1.1.1 -- 18 SEP 84 NEXT OBJECT ADDR: FD6A

ADDR DESCRIPTION/CONTENTS

ADDR DESCRIPTION/CONTENTS

FD05 Set Y = 0  
 FD06 3 pages of code to copy  
 FD08 ---  
 FD09 copy quit code handler to \$1000  
 FD17 Restore zero page to original state  
 FD23 enable HIGH RAM BANK1 (C08B)  
 FD26 (MLI) (C08B)  
 FD2B point RESET vector at \$1000 (03F2)  
 FD33 set power-up byte properly  
 FD38 go to quit code handler at \$1000 >>1000

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

FD3B \*\*\*\*\* NEW ROUTINE \*\*\*\*\*  
 THE ADDRESS OF THIS ROUTINE IS AT \$3EA.  
 WE COULD NOT DETERMINE ITS PURPOSE.

FD3B \*\*\*\*\* NEW ROUTINE \*\*\*\*\*  
 THE ADDRESS OF THIS ROUTINE IS AT \$3EA.  
 WE COULD NOT DETERMINE ITS PURPOSE.

FD3C get current P-reg in accumulator  
 FD3D save current P-reg  
 FD3E clear overflow flag  
 FD3F interrupts disabled?  
 FD41 no >>FD46  
 FD43 yes, set overflow flag (FD64)  
 FD46 disable interrupts  
 FD47 enable RAM, BANK2 (C083)  
 FD4D set carry, indicating error  
 FD4E pass a 5 to page 3 subroutine  
 FD50 call a page 3 subroutine <03D6>  
 FD53 store error number (BF0F)  
 FD56 enable RAM, BANK1 (C08B)  
 FD5C restore original P-reg  
 FD5E if error number is zero, (BF0F)  
 FD61 then indicate no error; >>FD64  
 FD63 otherwise indicate error  
 FD64 RETURN

FD65 \*\*\*\*\*  
 \* DATA AREA \*  
 \*\*\*\*\*  
 \*\*\*\*\* MLI COMMAND TABLE \*\*\*\*\*  
 IN HASH CODE ORDER: IF COMMAND IS...  
 ABCD EFGH (IN BINARY BITS)  
 INDEX IS COMPUTED AS:  
 000D EFGH  
 +0000 ABCD

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

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

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

NEXT OBJECT ADDR: FD9A

ADDR DESCRIPTION/CONTENTS

```

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

```

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

```

FDA5 CREATE
FDA7 DESTROY
FDA9 RENAME
FDAB SET FILE INFO
FDAD GET FILE INFO
FDAF ON LINE
FDB1 SET PREFIX
FDB3 GET PREFIX
FDB5 OPEN
FDB7 NEWLINE
FDB9 READ
FDBB WRITE
FDBD CLOSE
FDBF FLUSH
FDC1 SET MARK
FDC3 GET MARK
FDC5 SET EOF
FDC7 GET EOF
FDC9 SET BUF
FDCB GET BUF

```

FDCD \*\*\*\*\* MLI COMMAND INFO BYTE \*\*\*\*\*

```

PATHNAME FLAG
| REFERENCE NUMBER FLAG
| | DATETIME STAMP FLAG
| | | COMMAND NUMBER
| | | |
FDCD 1 0 1 - 00
FDCE 1 0 1 - 01
FDCF 1 0 1 - 02
FDD0 1 0 1 - 03
FDD1 1 0 0 - 04
FDD2 0 0 0 - 05
FDD3 0 0 0 - 06
FDD4 0 0 0 - 07
FDD5 1 0 0 - 08

```

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

NEXT OBJECT ADDR: FDD6

ADDR DESCRIPTION/CONTENTS

```

FDD6 0 1 0 - 09
FDD7 0 1 0 - 0A
FDD8 0 1 0 - 0B
FDD9 0 0 1 - 0C
FDDA 0 0 1 - 0D
FDEB 0 1 0 - 0E
FDDC 0 1 0 - 0F
FDDD 0 1 0 - 10
FDDF 0 1 0 - 11
FDDF 0 1 0 - 12
FDE0 0 1 0 - 13

```

FDE1 \*\*\*\*\* CONSTANTS - DATA AREA \*\*\*\*\*

```

FDE1 Blocks Used
FDE3 End of File
FDE6 Special ID (Must be 5 bits on)
FDE7 'HUSTON!' Author's name
FDEE Previous Block of Vol Dir Key Block

```

---

```

THE FOLLOWING IS COPIED TO SUBDIR HDR+$20
FDF0 Version of ProDOS
FDF1 Minimum Version
FDF2 Access Byte (D|Rn|B|000|W|R)
FDF3 Entry Length
FDF4 Entries per Block
FDF5 File Count
FDF7 Parent LSB (copied to SUBDIR HDR +$20)

```

---

```

FDF8 File Type (Directory)
FDF9 Block Number
FDFB Number of Blocks
FDFD End of File

```

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

```

FE00 100000000
FE01 010000000
FE02 001000000
FE03 000100000
FE04 000010000
FE05 000001000
FE06 000000100
FE07 000000001

```

FE08 \*\*\*\*\* OFFSETS TO DATA AT \$F300 \*\*\*\*\*

ProDOS MLI -- V1.1.1 -- 18 SEP 84 NEXT OBJECT ADDR: FE08  
-----  
ADDR DESCRIPTION/CONTENTS  
-----

FE08 Key Block  
FE0A # Blocks Used  
FE0C End of File  
FE0F \*\*\*\*\* SET/GET FILE\_INFO OFFSETS \*\*\*\*\*  
FE0F Access  
FE10 File Type  
FE11 Aux Type  
FE13 Storage Type  
FE14 Blocks Used (MSB on means GET only no SET)  
FE16 Datetime (Last Mod)  
FE1A Datetime (Creation)  
FE1E \*\*\*\*\* FATAL ERROR MESSAGE \*\*\*\*\*  
FE1E INSERT SYSTEM DISK AND RESTART  
FE46 ---  
FE46 \*\*\*\*\* VARIABLES - DATA AREA \*\*\*\*\*  
FE46 Parent Pointer Block  
FE48 Parent Entry Number  
FE49 Parent Entry Length  
FE4A Datetime (Creation)  
FE4E Version  
FE4F Min Version  
FE50 Access Byte  
FE51 Entry Length  
FE52 Entries per Block  
FE53 File Count  
FE55 Bit Map Pointer  
FE57 Total Blocks  
THE FOLLOWING 6 BYTES UNIQUELY IDENTIFY  
A FILE:  
FE59 Device Number  
FE5A Current Directory Block Number (HDR)  
FE5C Block Number of File Entry in Directory  
FE5E File Entry Number in Directory

ProDOS MLI -- V1.1.1 -- 18 SEP 84 NEXT OBJECT ADDR: FE5F  
-----  
ADDR DESCRIPTION/CONTENTS  
-----

FE5F \*\*\*\*\* FILE ENTRY BUFFER \*\*\*\*\*  
FE5F Type/Length (TTTTLLLL)  
FE60 File Name (Max 15) >>000F  
FE6F File Type  
FE70 Key Pointer  
FE72 Blocks Used  
FE74 End of File  
FE77 Datetime (Creation)  
FE7B Version  
FE7C Min Version  
FE7D Access Attribute  
FE7E Aux Type (Load Address/Record Length)  
FE80 Datetime (Last Mod)  
FE84 Header Pointer  
FE86 \*\*\*\*\* Variable Work Area \*\*\*\*\*  
FE86 3 Byte Scratch  
FE89 ---  
FE8A End of File  
FE8D Previous Mark  
FE90 Compare Vol Name Scratch  
FE91 Offset into VCB Table (\$D900)  
FE92 Offset into FCB Table (\$D800)  
FE93 Free FCB found Flag  
FE94 Number of Free Blocks needed  
FE96 Storage Type  
FE97 Number of Entries Examined or..  
FE97 FCB already open flag  
FE98 File Count  
FE9A Entries/Block Loop Count/Free FCB's refnum  
Free Entry Found Flag (if > 0) or..  
# of 1st bitmap block with free bit on or..  
bit for free  
FE9B # Blocks in Bitmap left to search  
FE9C Y Register temp  
FE9D Pathname Length  
FE9E Devnum for Prefix Directory Header  
FE9F Block of Prefix Directory Header  
FEA0 Bitmap Byte Offset in Page  
FEA2 Bitmap Page Offset  
FEA3 Bitmap Buffer Page (0 or 1)  
FEA4

ProDOS MLI -- V1.1.1 -- 18 SEP 84      NEXT OBJECT ADDR: FEA5  
 -----  
 ADDR    DESCRIPTION/CONTENTS  
 -----

FEA5 Bitmap Flag (if \$80, needs writing)  
 FEA6 Bitmap DEVNUM  
 FEA7 Bitmap Block Number  
 FEA9 Bitmap Block offset for Multiblock Bitmaps

    New Mark to be Positioned to for Set Mark  
     or New Moving Mark (for READ)  
 FEAA or New EOF for SET\_EOF

FEAD Request Count (Read/Write etc.)  
 FEAF Multi-Block I/O count  
 FEB0 Newline character  
 FEB1 Newline mask  
 FEB2 I/O Transfer occurred flag  
 FEB3 MLI Command \* 2  
 FEB4 Ored into Access Flags (\$20 - Backup)  
 FEB5 Duplicate Volume Flag (if \$FF)  
 FEB6 Duplicate Volume's VCB index  
 FEB7 MLI function code (low 5 bits)

    Characters in current Pathname indx lvl or  
 FEB8 ONLINE: volname len - loop index  
 FEB9 new pathname: index to last name  
     old pathname: index to last name or..

FEBA ONLINE: index to Data buffer  
 FEBB Old PFXPTR value  
 FEBC Pathname fully qualified flag (if \$FF)  
     Pathname: temp save area for index or..

FEBD ONLINE: DEVCNT  
 FEBE close-all error code  
 FEBF Set EOF: new Key Block pointer  
 FEC1 New storage type (SET\_EOF)  
 FEC2 Freed Blocks count  
 FEC4 EOF Block number (MSB then LSB)  
 FEC6 EOF byte offset into Block  
 FEC8 EOF - Master index counter  
 FEC9 Save area for index into table below

FECA \*\*\*\*\* DEVICE TABLE BUILT BY ONLINE \*\*\*\*\*  
 (also used by SET\_EOF to keep track of  
 8 blocks to be freed at a time)

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

ProDOS MLI -- V1.1.1 -- 18 SEP 84      NEXT OBJECT ADDR: FEF5  
 -----  
 ADDR    DESCRIPTION/CONTENTS  
 -----

FEF7 \*\*\*\*\* \$FEF7-\$FEFF NOT USED \*\*\*\*\*  
 FEF7 not used



**ProDOS SYSTEM GLOBAL PAGE--MLI Global Page**

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

ProDOS System Global Page	Next Object Address: BF00
ADDR	CONTENTS

**Jump Vectors**

BF00-BF02	ENTRY	JMP to MLI.
BF03-BF05	JSPARE	JMP to system death code (via \$BFF6).
BF06-BF08	DATETIME	JMP to Date/Time routine (RTS if no clock).
BF09-BF0B	SYSERR	JMP to system error handler.
BF0C-BF0E	SYSDEATH	JMP to system death handler.
BF0F	SERR	System error number.
		<b>Device Information</b>
BF10-BF11	DEVADR01	Slot 0 reserved
BF12-BF13	DEVADR11	Slot 1, drive 1 device driver address.
BF14-BF15	DEVADR21	Slot 2, drive 1 device driver address.
BF16-BF17	DEVADR31	Slot 3, drive 1 device driver address.
BF18-BF19	DEVADR41	Slot 4, drive 1 device driver address.
BF1A-BF1B	DEVADR51	Slot 5, drive 1 device driver address.
BF1C-BF1D	DEVADR61	Slot 6, drive 1 device driver address.
BF1E-BF1F	DEVADR71	Slot 7, drive 1 device driver address.
BF20-BF21	DEVADR02	Slot 0 reserved.
BF22-BF23	DEVADR12	Slot 1, drive 2 device driver address.
BF24-BF25	DEVADR22	Slot 2, drive 2 device driver address.
BF26-BF27	DEVADR32	/RAM device driver address (need extra 64K).
BF28-BF29	DEVADR42	Slot 4, drive 2 device driver address.
BF2A-BF2B	DEVADR52	Slot 5, drive 2 device driver address.
BF2C-BF2D	DEVADR62	Slot 6, drive 2 device driver address.
BF2E-BF2F	DEVADR72	Slot 7, drive 2 device driver address.
BF30	DEVNUM	Slot and drive (DSSS0000) of last device.
BF31	DEVCNT	Count (minus 1) of active devices.
BF32-BF3F	DEVLIST	List of active devices (slot, drive and identification--DSSSIIII).
BF40-BF4F	IRQITX	Copyright notice.
BF50-BF55		Switch in language card and call IRQ handler at \$FFD8.
BF56-BF57	TEMP	Temporary storage for IRQ code.
BF58-BF6F	BITMAP	Bitmap of low 48K of memory.
BF70-BF71	BUFFER1	Open file 1 buffer address.
BF72-BF73	BUFFER2	Open file 2 buffer address.
BF74-BF75	BUFFER3	Open file 3 buffer address.
BF76-BF77	BUFFER4	Open file 4 buffer address.
BF78-BF79	BUFFER5	Open file 5 buffer address.
BF7A-BF7B	BUFFER6	Open file 6 buffer address.
BF7C-BF7D	BUFFER7	Open file 7 buffer address.
BF7E-BF7F	BUFFER8	Open file 8 buffer address.

ProDOS System Global Page      NEXT OBJECT ADDRESS: BF80  
 -----  
 ADDR      LABEL      CONTENTS

**BF80-BF81**    INTRUPT1    **Interrupt Information**  
 Interrupt handler address (highest priority).  
**BF82-BF83**    INTRUPT2    Interrupt handler address.  
**BF84-BF85**    INTRUPT3    Interrupt handler address.  
**BF86-BF87**    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-BF8F**    INTADDR     Interrupt return address.  
**BF90-BF91**    DATE        **General System Info**  
 YYYYYYMM MMMDDDDD.  
**BF92-BF93**    TIME        ...HHHH ...MMMMMM.  
**BF94**        LEVEL       Current file level.  
**BF95**        BUBIT       Backup bit.  
**BF96-BF97**    SPARE1      Currently unused.  
**BF98**        MACHID      Machine ID byte.  
              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  
              ..00 ..00    Unused  
              ..01 ..00    48K  
              ..10 ..00    64K  
              ..11 ..00    128K  
              .... X...    Reserved  
              .... 00.    No 80-column card  
              .... 01.    80-column card present  
              .... 10    No compatible clock  
              .... 11    Compatible clock present  
**BF99**        SLTBYT      Slot ROM map (bit on indicates ROM present).  
**BF9A**        PFIXPTR     Prefix flag (0 indicates no active prefix).  
**BF9B**        MLIACTV     MLI active flag (1... .. indicates active).  
**BF9C-BF9D**    CMDADR      Last MLI call return address.  
**BF9E**        SAVEX       X-register savearea for MLI calls.  
**BF9F**        SAVEY       Y-register savearea for MLI calls.

ProDOS System Global Page      NEXT OBJECT ADDRESS: BFA0  
 -----  
 ADDR      LABEL      CONTENTS

**BFA0-BFCF**    **Language Card Bank Switching Routines**  
 Language card entry and exit routines.  
**BFA0**        EXIT  
**BFAA**        EXIT1  
**BFB5**        EXIT2  
**BFB7**        MLIENF1  
**BFD0-BFF3**    **Interrupt Routines**  
 Interrupt entry and exit routines.  
              BFD0  
              BFDE    IROXIT1  
              BFE2    IROXIT2  
              BFE7    ROMXIT  
              BFEB    IRQENT  
**BF4**        BNKBYT1    **Data**  
**BF5**        BNKBYT2    Storage for byte at \$E000.  
**BF6-BFB**    Storage for byte at \$D000.  
 Switch on language card and call system death handler (\$D1E4).  
**BFFC**        IBAKVER    **Version Information**  
 Minimum version of Kernel needed for this interpreter.  
**BFFD**        IVERSION   Version number of this interpreter.  
**BFFE**        KBAKVER    Minimum version of Kernel compatible with this Kernel.  
**BFFF**        KVERSION   Version number of this Kernel.

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

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

1000    MODULE STARTING ADDRESS

```

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

1000    \*\*\*\*\* 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
C00F    Select alternate character set
C010    Keyboard Strobe
C082    ROM select
  
```

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

```

FC58    Home
FC9C    Clear to end of line
FD0C    Read a key
FD8E    Output a Carriage Return
FDED    Output a Character
FE89    Set Keyboard
FE93    Set Video
FF3A    Sound Bell
  
```

1000    \*\*\*\*\* INITIALIZATION \*\*\*\*\*

```

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

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

```

1012    Mark pages $0, $1, $4 through $7
1014    and $BF as in use
  
```

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

```

1027    Clear Screen and Home cursor <FC58>
102A    Go down 1 line <FD8E>
102D    Get Pointer to Prompt1 (Prefix)
102F    and store it in Print Routine (11E9)
1037    Call Print Routine <11E6>
103A    Position to line 3
1041    Call MLI (GET_PREFIX) <BF00>
1044    Data: GET_PREFIX command number
1045    Data: Pointer to Parameter list
1047    Terminate Prefix with 0 (0280)
104A    for Print routine
104F    Get Pointer to Prefix
1051    and store it in Print Routine (11E9)
1059    And Print it <11E6>
  
```

105C    \*\*\*\*\* GET\_PREFIX NAME \*\*\*\*\*

```

105C    Initialize counter
1063    Read a key <FD0C>
1066    Is it CARRIAGE RETURN?
1068    Yes, then accept Prefix >>10B8
106A    No, then save character
106B    Clear to end of line <FC9C>
106E    Retrieve character
106F    Is it ESCAPE?
1071    Yes, then start all over again >>1027
1073    Is it CANCEL?
1075    Yes, then start all over again >>1027
1077    Is it TAB?
1079    Yes, then sound Bell, get another character >>108E
107B    Is it BACKSPACE?
107D    No, then keep checking >>108C
107F    Yes, then is there room to move back?
1081    No, then don't try >>1086
  
```

ProDOS QUIT Code -- V1.1.1 -- 18 SEP 84      NEXT OBJECT ADDR: 1083  
 ADDR    DESCRIPTION/CONTENTS

1083 Decrement cursor horizontal position  
 1085 Decrement counter  
 1086 Clear to end of line <FC9C>  
 1089 Try again >>1063  
 108C Continue if greater than or equal to BACKSPACE >>1094  
 108E Else, sound Bell <FF3A>  
 1091 Try again >>1063  
 1094 Is it less than or equal to "Z"?  
 1096 Yes, keep checking >>109A  
 1098 Turn off lowercase  
 109A Is it less than ".", "?"  
 109C Yes, Invalid - try again >>108E  
 109E Is it greater than "Z"?  
 10A0 Yes, Invalid - try again >>108E  
 10A2 Is it less than or equal to "9"?  
 10A4 Yes, keep checking >>10AA  
 10A6 Is it less than "A"?  
 10A8 Yes, Invalid - try again >>108E  
 10AA Else, valid character - increment counter  
 10AB Found 39 characters  
 10AD Yes, then start all over >>1075  
 10AF Put valid character in buffer (0280)  
 10B2 and Print it <FDED>  
 10B5 Go back for more >>1063  
 10B8 Check counter  
 10BA If 0 then go on >>105E  
 10BC Else, save length (0280)  
 10BF Call MLI (SET PREFIX) <BF00>  
 10C2 Data: SET PREFIX command number  
 10C3 Data: Pointer to Parameter list  
 10C5 Carry on if no error >>105E  
 10C7 Sound Bell <FF3A>  
 10CA Force branch to  
 10CC always be taken >>1075

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

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

ProDOS QUIT Code -- V1.1.1 -- 18 SEP 84      NEXT OBJECT ADDR: 10F9  
 ADDR    DESCRIPTION/CONTENTS

10F9 Is it ESCAPE?  
 10FB No, keep checking >>1103  
 10FD Yes, get Cursor horizontal position  
 10FF If not 0 try again >>10CE  
 1101 If 0 start all over again >>10CC  
 1103 Is it CANCEL?  
 1105 Yes, try again >>10CE  
 1107 Is it TAB?  
 1109 Yes, sound Bell - try again >>1114  
 110B Is it BACKSPACE?  
 110D No, keep checking >>1112  
 110F Yes, then handle it >>11D0  
 1112 Continue if greater than or equal to BACKSPACE >>111A  
 1114 Sound Bell <FF3A>  
 1117 Go back and try again >>10EA  
 111A Is it CARRIAGE RETURN?  
 111C Yes, then go load Application >>1147  
 111E Is it less than or equal to "Z"?  
 1120 Yes, keep checking >>1124  
 1122 Turn off lower case  
 1124 Is it less than ".", "?"  
 1126 Yes, Invalid - try again >>1114  
 1128 Is it greater than "Z"?  
 112A Yes, Invalid - try again >>1114  
 112C Is it less than or equal to "9"?  
 112E Yes, keep checking >>1134  
 1130 Is it less than "A"?  
 1132 Yes, Invalid - try again >>1114  
 1134 Else, valid character - save it  
 1135 Clear to end of line <FC9C>  
 1138 Retrieve character  
 1139 and Print it <FDED>  
 113C Increment counter  
 113D Found 39 characters?  
 113F Yes, start again >>1105  
 1141 No, save character in buffer (0280)  
 1144 and go get another >>10EA

1147 \*\*\*\*\* LOAD AND EXECUTE APPLICATION \*\*\*\*\*

1147 Output a blank  
 114C Store length of Application name (0280)  
 114F Call MLI (GET FILE INFO) <BF00>  
 1152 Data: GET\_FILE\_INFO command number  
 1153 Data: Pointer to Parameter list  
 1155 Continue if no error >>115A  
 1157 Else, go to Error Handler >>11F6

```

ProDOS QUIT Code -- V1.1.1 -- 18 SEP 84      NEXT OBJECT ADDR: 1157
-----
ADDR  DESCRIPTION/CONTENTS
-----
115A Get File Type (12D5)
115D Is it ProDOS System file?
115F Yes, continue >>1166
1161 No, indicate Error $01
1163 Go to Error Handler >>11F6

1166 Set Reference number to 0
116B Call MLI (CLOSE) <BF00>
116E Data: CLOSE command number
116F Data: Pointer to Parameter list
1171 Continue if no error >>1176
1173 Else, go to Error Handler >>11F6
1176 Get Access Byte (12D4)
117B Yes, >>1182
117D No, Indicate Error $27
117F Go to Error Handler >>11F6

1182 Call MLI (OPEN) <BF00>
1185 Data: OPEN command number
1186 Data: Pointer to Parameter list
1188 Continue if no error >>118D
118A Else, go to Error Handler >>11F6

118D Get Reference Number (12E8)
1190 and update READ and (12EC)
1193 GET_EOF parameter lists (12F4)
1196 Call MLI (GET_EOF) <BF00>
1199 Data: GET_EOF_command number
119A Data: Pointer to Parameter list
119C Continue if no error >>11A1
119E Else, go to Error Handler >>11F6

11A1 Is EOF mark less than $10000 (12F7)
11A4 Yes, continue on >>11AB
11A6 No, Indicate Error $27
11A8 Go to Error Handler >>11F6

11AB Transfer EOF to Request count (12F5)
11AE in READ parameter list (12EF)
11B7 Call MLI (READ) <BF00>
11BA Data: READ command number
11BB Data: Pointer to Parameter list
11BD Save status of READ
11BE Call MLI (CLOSE) <BF00>
11C1 Data: Get Prefix command number
11C2 Data: Pointer to Parameter list
11C4 Continue if no error >>11CA
11C6 Else, retrieve status
11C7 and go to Error Handler >>11F6

ProDOS QUIT Code -- V1.1.1 -- 18 SEP 84      NEXT OBJECT ADDR: 11C7
-----
ADDR  DESCRIPTION/CONTENTS
-----
11CA Was READ good?
11CB No, go to Error Handler >>11C7
11CD Yes, execute application >>2000

11D0 ***** BACKSPACE ROUTINE *****
11D0 Get cursor position horizontal
11D2 If 0 exit routine >>11E3
11D4 Decrement counter
11D5 Output a space
11DA Move cursor back 2 spaces
11DE Output a space <FDED>
11E1 Move cursor back 1 space
11E3 Return to get another character >>10EA

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

11F6 ***** PRINT ERROR MESSAGE *****
11F6 Save Accumulator (Error Number)
11F8 Position to line 12
11FF Get Error number
1201 Is it $01?
1203 NO, then keep checking >>1211
1205 Get Pointer to Error1 (Not System file)
1207 and store it in Print Routine (11E9)
120F Branch always taken >>1237
1211 Is it $40?
1213 Yes, then indicate Error3 >>122D
1215 Is it $44?
1217 Yes, then indicate Error3 >>122D
1219 Is it $45?
121B Yes, then indicate Error3 >>122D
121D Is it $46?
121F Yes, then indicate Error3 >>122D
1221 Else, Get Pointer to Error2 (I/O Error)
1223 and store it in Print Routine (11E9)
122B Branch always taken >>1237
122D Get Pointer to Error3 (Path not found)
122F and store it in Print Routine (11E9)
1237 Print Error message <11E6>
123A Position to line 0

```

ProDOS QUIT Code -- V1.1.1 -- 18 SEP 84      NEXT OBJECT ADDR: 123E  
 -----  
 ADDR    DESCRIPTION/CONTENTS  
 -----

ProDOS QUIT Code -- V1.1.1 -- 18 SEP 84      NEXT OBJECT ADDR: 12EA  
 -----  
 ADDR    DESCRIPTION/CONTENTS  
 -----

123E    Return to Get Application code >>10D1

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

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

    Prompt2  
 1269    'ENTER PATHNAME OF NEXT APPLICATION'

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

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

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

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

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

    OPEN Parmlist

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

    CLOSE Parmlist

12F9    Parmcount  
 12EA    Reference Number

    READ Parmlist

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

    GET\_EOF Parmlist

12F3    Parmcount  
 12F4    Reference Number  
 12F5    EOF Mark

    GET\_SET\_PREFIX Parmlist

12F8    Parmcount  
 12F9    Pathname

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

12FB    These unused bytes are \$D3FB-\$D3FF in high RAM  
 12FF    and \$59FB-\$59FF when loaded as part of "PRODOS" file.

Disk II Device Driver -- V1.1.1 -- 18 SEP 84 NEXT OBJECT ADDR: D000  
 ADDR DESCRIPTION/CONTENTS

D000 MODULE STARTING ADDRESS  
 \*\*\*\*\*  
 \* 5.25" DISK DEVICE DRIVER \*  
 \* \* \* \* \*  
 \* RESIDES AT \$D000-\$D6FF \*  
 \* \* \* \* \*  
 \* VERSION 1.1.1 -- 18 SEP 84 \*  
 \* \* \* \* \*

D000 \*\*\*\*\* ZERO PAGE EQUATES \*\*\*\*\*  
 Checksum  
 Workbyte  
 Slot (Temporary)  
 Command  
 Unit Number  
 I/O Buffer Pointer (low)  
 I/O Buffer Pointer (high)  
 Block Number (low)  
 Block Number (high)

D000 \*\*\*\*\* INTERNAL EQUATES \*\*\*\*\*  
 Dummy Block Buffer (1st half)  
 Dummy Block Buffer (2nd half)  
 D000 \*\*\*\*\* EXTERNAL EQUATES \*\*\*\*\*  
 Phase Zero Off  
 Motor Off  
 Motor On  
 Drive Select  
 Read Data Register  
 Write Data Register  
 Set Read Mode  
 Set Write Mode  
 Read Data Register (slot 6)

D000 \*\*\*\*\* 5.25" DISK DRIVER ENTRY \*\*\*\*\*  
 D000 Clear decimal mode  
 D001 Clear phases in case IWM device in this slot <D6BE>  
 D004 Five NOP's so code below will  
 D005 fit up against Table at \$D196  
 D009 Check validity of calling parameters <D6D0>  
 D00C If not valid exit with error >>D034  
 D00E Convert Block Number to a Track and Sector

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

D010 ---  
 D014 0000000T TTTTABC  
 D015 . . . >>D010  
 D017 . . . >>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" DSSS0000  
 D042 Strip out Drive 0SSS0000  
 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 -- V1.1.1 -- 18 SEP 84 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  
 D0DF 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 EME 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 -- V1.1.1 -- 18 SEP 84 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 -- V1.1.1 -- 18 SEP 84 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>  
 D16E Get delay value from table (D373)  
 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 - 00000PPC  
 D18D Merge in slot number - 0SSS0PPP  
 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 Prenibblize  
 Bit mask Tables and Eplilog Table in  
 unused areas

D196 Read Translate  
 Bit Mask 1  
 D1A0 00000000  
 D1A1 10000000  
 D1A2 01000000  
 D1A3 11000000

Read Translate  
 Bit Mask 2  
 D1C0 00000000  
 D1C1 00100000  
 D1C2 00010000  
 D1C3 00110000

Eplilog Table (\$DE,\$AA,\$EB)

Disk II Device Driver -- V1.1.1 -- 18 SEP 84 NEXT OBJECT ADDR: D1C4  
 ADDR DESCRIPTION/CONTENTS

Read Translate  
 Bit Mask 3  
 D1E0 00000000  
 D1E1 00001000  
 D1E2 00000100  
 D1E3 00001100

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

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

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

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

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

D356 Track number  
 D357 Sector number  
 D358 Error number

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

Disk II Device Driver -- V1.1.1 -- 18 SEP 84 NEXT OBJECT ADDR: D367  
 -----  
 ADDR DESCRIPTION/CONTENTS  
 -----

D369 Retry count (initially 64)  
 D36A Recalibration count (initially 1)  
 D36B Counter for Read Address routine  
 D36B Temporary storage for Read Address routine  
 D36B Track counter for Arm Move routine  
 D36C Checksum computation  
 D36D Volume found  
 D36E Sector found  
 D36F Delay counter (low byte)  
 D370 Track found  
 D370 Checksum found  
 D371 Delay counter (high byte)  
 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 to \$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 -- V1.1.1 -- 18 SEP 84 NEXT OBJECT ADDR: D3C4  
 -----  
 ADDR DESCRIPTION/CONTENTS  
 -----

D3C4 Initialize checksum  
 D3C9 Read "odd" encoded byte LX1X1X1X (C08C)  
 D3CE Align "odd" bits X1X1X1X1  
 D3CF Save for later (D36B)  
 D3D2 Read "even" encoded byte LX1X1X1X (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 -- V1.1.1 -- 18 SEP 84 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 XXXXX00  
 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 XXXXX00  
 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 XXXXX00  
 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 is 1st trailing mark (\$DE)?

Disk II Device Driver -- V1.1.1 -- 18 SEP 84 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  
 D4F2 Get Unit number DSSS000  
 D4F4 Divide by 16 0000DSSS  
 D4F8 Put Drive into Carry 0000DSSS D  
 D4FA Strip out Drive 0000SSSS D  
 D4FC Roll left 0000SSSS  
 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 into zero page for timing  
 D511 Use \$FF for "sync" byte  
 D513 Write first "sync" byte (C08F)  
 D519 Set counter for four more  
 D51C Delay so that writes occur  
 D51D Exactly on 40 cycle loops

Disk II Device Driver -- V1.1.1 -- 18 SEP 84 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?  
 D56E 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 -- V1.1.1 -- 18 SEP 84 NEXT OBJECT ADDR: D59D  
 ADDR DESCRIPTION/CONTENTS

D59D Put result in X-reg for table lookup  
 D59E Lookup "disk byte" in table (D203)  
 D5A1 Get slot  
 D5A3 Write "disk byte" (C08D)  
 D5A9 Get data byte (Primary buffer - page 2) (1100)  
 D5AC Increment offset - Done yet?  
 D5AD No, then do another >>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 -- V1.1.1 -- 18 SEP 84 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  
 D62A Get current value from stack 0000BA00  
 D62B Merge in new bits using table 00DCBA00 (D1C0)  
 D62E Save result on stack  
 D62F Get data byte (top third) XXXXXXXX (10AC)  
 D632 Get last two bits 000000EF  
 D634 Put in X-reg for table lookup  
 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 -- V1.1.1 -- 18 SEP 84 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 \*\*\*\*\*  
 D6D0 Check command code  
 D6D2 Is it greater or equal to 4?  
 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 -- V1.1.1 -- 18 SEP 84      NEXT OBJECT ADDR: FF9B  
 -----  
 ADDR    DESCRIPTION/CONTENTS

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

FF9B    \*\*\*\*\* GLOBAL PAGE EQUATES \*\*\*\*\*  
 BF56    Temporary storage 1  
 BF57    Temporary storage 2  
 BF88    A register savearea  
 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)  
 FFA1    Replace \$45 with A-Register  
 FFA2    since it may have been destroyed  
 FFA4    Load Status register  
 FFA5    Restore onto stack  
 FFA6    Isolate B flag - Was it a BRK?  
 FFA8    Yes, skip Interrupt stuff >>FFC2  
 FFAD    Else, Check location \$D000 (D000)  
 FFAF    Do we have RAM active  
 FFBB    Yes, indicate so >>FFB3  
 FFB1    Else, indicate ROM  
 FFB3    Update Bank ID byte (BF8D)  
 FFB6    Also save temporarily (BF57)  
 FFB9    Push (\$BF50) address of  
 FFBF    routine to bank in Ram and  
 FFBF    call IRQ on the stack  
 FFBF    Push a new P-Register on stack with  
 FFC1    the Interrupt Disable flag set  
 FFC2    Push (\$FA41) address less 1 of  
 FFC4    Monitor IRQ on the stack

IRQ Handler -- V1.1.1 -- 18 SEP 84      NEXT OBJECT ADDR: FFC8  
 -----  
 ADDR    DESCRIPTION/CONTENTS

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 (BF98)  
 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 \*\*\*\*\*  
 FFE8    These unused bytes are at \$FEC-\$4F9 when  
 FFE9    loaded as part of the "PRODOS" file.  
 FFFA    \*\*\*\*\* VECTORS \*\*\*\*\*  
 FFFA    NMI Vector  
 FFFC    Reset Vector  
 FFFE    IRQ Vector

HOW "BASIC.SYSTEM" IS LOADED AND RELOCATED

- 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          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 -- V1.1.1 -- 18 JUN 84 NEXT OBJECT ADDR: 2000  
ADDR DESCRIPTION/CONTENTS

BI Relocator -- V1.1.1 -- 18 JUN 84 NEXT OBJECT ADDR: 2000  
ADDR DESCRIPTION/CONTENTS

2000 MODULE STARTING ADDRESS

\*\*\*\*\* BASIC GLOBAL PAGE \*\*\*\*\*  
\* \* PRODOS BASIC INTERPRETER RELOCATOR \* \*  
\* \* LOADED AS THE FIRST TWO BLOCKS \* \*  
\* \* OF BASIC.SYSTEM AT \$2000. \* \*  
\* \* THIS ROUTINE MOVES THE BASIC \* \*  
\* \* INTERPRETER TO \$9A00-\$BCFF. \* \*  
\* \* FOR PRODOS VERSION 1.1.1 \* \*  
\* \* (BASIC Version Number is 1.1, \* \*  
\* \* Modify date is 18 JUN 84) \* \*  
\*\*\*\*\*

\*\*\*\*\* BASIC GLOBAL PAGE \*\*\*\*\*  
BASIC INTERPRETER VERSION NUMBER  
BASIC INTERPRETER ENTRY POINT  
BI COMMAND SCANNER (SYNTAX)  
COUT VECTORS FOR EACH SLOT  
KSWL VECTORS FOR EACH SLOT  
DEFAULT SLOT NO.  
BE3D DEFAULT DRIVE NO.  
BEFF HIMEM

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

"FROM" POINTER FOR COPY  
"TO" POINTER FOR COPY  
CSWL VECTOR  
KSWL VECTOR  
APPLESOFT START OF STRINGS  
APPLESOFT HIMEM  
APPLESOFT TRACE FLAG

\*\*\*\*\* SYSTEM GLOBAL PAGE \*\*\*\*\*  
MACHINE LANGUAGE INTERFACE ENTRY  
LAST DEVICE USED  
MEMORY MAP  
MACHINE TYPE FLAGS  
SLOTS WHICH CONTAINS CARDS WITH ROM  
IF 0, NO PREFIX ACTIVE  
INTERPRETER VERSION NUMBER

\*\*\*\*\* EXTERNAL ADDRESSES \*\*\*\*\*

PATHNAME BUFFER  
PREFIX BUFFER  
START OF PREFIX NAME  
WARMSTART VECTOR  
COLDSTART VECTOR  
BRK HANDLER ADDRESS  
RESET HANDLER ADDRESS  
POWER-UP BYTE  
APPLESOFT & VECTOR  
CTL-Y VECTOR

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

FIRST SCREEN BUFFER LINE  
SCREEN BUFFER LINE  
SCREEN BUFFER LINE

0000  
0001  
0002  
0003  
0036  
0038  
006F  
0073  
00F2

0200  
0280  
0281

03D0  
03D3  
03F0  
03F1  
03F2  
03F3  
03F4  
03F5  
03F8

0400  
0480  
0628

BF00  
BF30  
BF58  
BF98  
BF99  
BF9A  
BFFD

\*\*\*\*\* ROM ADDRESSES \*\*\*\*\*  
APPLESOFT ENTRY POINT  
BRK HANDLER  
INIT SCREEN, MONITOR, ETC.  
CLEAR SCREEN, HOME CURSOR  
STANDARD CHARACTER OUT  
CHARACTER OUTPUT TO SCREEN  
SET NORMAL CHARACTER ATTRIBUTE

2000  
2006  
2007  
200E

\*\*\*\*\* BASIC INTERP RELOCATOR ENTRY \*\*\*\*\*  
JUMP OVER STARTUP FILENAME >>2047  
STARTUP FILENAME LENGTH (7)  
'STARTUP'  
ALLOW FOR 64 CHAR FILENAME

2047 \$00 --> \$2400  
204B \$02 --> \$9A00  
2055 COPY 35 PAGES  
2058 COPY INTERP TO HIGH MEMORY AT \$9A00 <20C4>  
205D PAGE FOLLOWING INTERP IMAGE IS...  
205F BASIC GLOBAL PAGE IMAGE  
2061 COPY THAT TO \$BE00 <20C4>  
2064 TO GET 40-COL DISPLAY, SEND A CTRL-U  
2066 OUT THE NORMAL OUTPUT VECTOR. <FDED>  
2069 SET NORMAL CHARACTER ATTRIBUTE <FE84>  
206C INITIALIZE SCREEN/WINDOW <FB2F>  
206F CLEAR SCREEN/HOME CURSOR <FC58>



BI Relocator -- V1.1.1 -- 18 JUN 84      NEXT OBJECT ADDR: 2076  
 ADDR    DESCRIPTION/CONTENTS

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 \*\*\*\*\* ERROR EXIT \*\*\*\*\*  
 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-->\$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)  
 2115 ---  
 211B SET HIMEM TO \$9600  
 211D IN VARIOUS PLACES  
 2124 GOT A DEFAULT PREFIX? (BF9A)

BI Relocator -- V1.1.1 -- 18 JUN 84      NEXT OBJECT ADDR: 2127  
 ADDR    DESCRIPTION/CONTENTS

2127 NO >>214E  
 2129 YES, MLI: GET PREFIX <BF00>  
 212F ERROR? >>218B  
 2136 BACKSCAN PREFIX FOR "/"'S (0280)  
 213B AND COUNT THEM IN \$21EE (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)  
 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

BI Relocator -- V1.1.1 -- 18 JUN 84      NEXT OBJECT ADDR: 21FF  
 -----  
 ADDR    DESCRIPTION/CONTENTS  
 -----

BI Relocator -- V1.1.1 -- 18 JUN 84      NEXT OBJECT ADDR: 22A5  
 -----  
 ADDR    DESCRIPTION/CONTENTS  
 -----

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

2400 \*\*\*\*\* START OF BI IMAGE \*\*\*\*\*  
 2400            BASIC INTERP IMAGE

\*\*\*\*\* 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

\*\*\*\*\* DATA \*\*\*\*\*  
 221A CSWL (2004) INTERCEPT ADDR  
 221C KSWL (2209) INTERCEPT ADDR  
 221E GET FILE INFO PARMLIST  
 221F FILE NAME IS AT \$2006  
 2221 15 BYTES RESERVED FOR OTHER GET\_FILE PARMS (NOT USED)  
 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

BASIC Interpreter (BI) -- V1.1.1 -- 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 V1.1.1 * * * * *
* * * * * *****
***** ZERO PAGE ADDRESSES *****

```

```

0024 CURSOR HORIZONTAL
0028 SCREEN LINE BASE ADDR
0029 MONITOR PROMPT CHARACTER
0033 CRT DISPLAY VECTOR (CSWL)
0036 KEYBOARD INPUT VECTOR (KSWL)
0037 SCRATCH POINTER AND LOOP COUNTER
0038 SCRATCH POINTER AND LOOP COUNTER
0039 SCRATCH POINTER AND LOOP COUNTER
003A POINTER TO APPLESOFT VARIABLES
003B APPLESOFT: LINE NUMBER
003C APPLESOFT: START OF PROGRAM PTR
003D APPLESOFT: LOMEM (START OF VARS)
003E APPLESOFT: START OF ARRAY VARS PTR
003F APPLESOFT: START OF FREEAREA PTR
0050 APPLESOFT: START OF STRINGS PTR
0051 APPLESOFT: HIMEM (END OF STRINGS)
0067 APPLESOFT: CURRENT LINE BEING EXECUTED
0068 APPLESOFT: ADDR OF LINE AFTER FINDLINE
006A
006B
006C
006E
006D
006F
0070
0073
0074
0075
0076
009B
009C

```

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

ADDR DESCRIPTION/CONTENTS

```

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 *****
START OF 6502 STACK
KEYBOARD INPUT LINE BUFFER
POWERON RESET FLAG

```

```

***** BI GLOBAL PAGE *****
EXTERNAL COMMAND ENTRY TO BI
PRINT ERROR MESSAGE ENTRY TO BI
PRODOS ERROR CODE
OUTPUT VECTORS FOR ALL SLOTS
CURRENT OUTPUT VECTOR
CURRENT INPUT VECTOR
PRODOS INTERCEPT VECTORS (INPUT/OUTPUT)
BI'S INTERNAL REDIRECTION VECTORS
DEFAULT SLOT
DEFAULT DRIVE
A REGISTER SAVE AREA
X REGISTER SAVE AREA
Y REGISTER SAVE AREA
TRACE FLAG (APPLESOFT TRACE ON/OFF)
IMMEDIATE COMMANDS=0, DEFERRED=1
EXEC FILE ACTIVE=$80
READ FILE ACTIVE=$80
WRITE FILE ACTIVE=$80
READING PREFIX ACTIVE=$80
DIRECTORY FILE BEING ACCESSED
FREE STRING SPACE DURING GARBAGE COLLECT
BUFFERED I/O BYTE COUNT
INDEX INTO INPUT COMMAND LINE
LAST OUTPUT CHAR TO PREVENT RECURSION
NUMBER OF OPEN NON-EXEC FILES
EXEC FILE BEING CLOSED FLAG
READ FILE IS TRANSLATED DIRECTORY
VECTOR TO EXTERNAL COMMAND HANDLER
LENGTH-1 OF EXTERNAL COMMAND STRING
COMMAND NUMBER
PARAMETERS ALLOWED FOR THIS COMMAND
(SEE BIT DEFINITIONS IN TABLE LATER)
PARAMETERS FOUND WITH THIS COMMAND

```

```

BE06
BE0C
BE0F
BE10
BE30
BE32
BE34
BE38
BE3C
BE3D
BE3E
BE3F
BE40
BE41
BE42
BE43
BE44
BE45
BE46
BE47
BE49
BE4A
BE4B
BE4C
BE4D
BE4E
BE4F
BE50
BE52
BE53
BE54
BE56

```

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

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

(SAME BIT DEFINITIONS AS FOR PBITS)

BE58 A KEYWORD VALUE  
 BE5A B KEYWORD VALUE  
 BE5D E KEYWORD VALUE  
 BE5F L KEYWORD VALUE  
 BE61 S KEYWORD VALUE  
 BE62 D KEYWORD VALUE  
 BE63 F KEYWORD VALUE  
 BE65 R KEYWORD VALUE  
 BE68 @ KEYWORD VALUE  
 BE6A T KEYWORD VALUE  
 BE6B SLOT NUMBER FROM IN# OR PR#  
 BE70 ISSUE 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 \*\*\*\*\*

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

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

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

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

C000 KEYBOARD STROBE  
 C010 KEYBOARD STROBE CLEAR  
 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, WARMDS)

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  
 9A1C KSWL/H ALREADY SET? <9F76>  
 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

9A2F \*\*\*\*\* OUTPUT INTERCEPT: MODE = 0 \*\*\*\*\*  
 (IMMEDIATE MODE)

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

BASIC Interpreter (BI) -- V1.1.1 -- 18 JUN 84 NEXT OBJECT ADDR: 9AA3

ADDR DESCRIPTION/CONTENTS

ADDR DESCRIPTION/CONTENTS

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

9AA3 \*\*\*\*\* SET CSWL/KSWL INTERCEPTS \*\*\*\*\*  
 9AA3 ---  
 9AA4 COPY VDOSIO VECTORS (BE34)  
 9AA7 TO CSWL  
 9AB1 AND KSWL  
 9AB9 EXIT TO CALLER

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, EXIT1  
 9A6E WE WERE'NT TRACING AFTER ALL, RESTORE X (BE3F)  
 9A71 AND A REGS, THEN FALL THRU TO EXIT (9F61)

9ABA \*\*\*\*\* INPUT INTERCEPT: MODE = 0 \*\*\*\*\*  
 (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

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

9AC5 NO EXEC FILE, RESTORE REAL CSWL/KSWL <9A00>  
 9AC8 NO, READ A KEY FROM KEYBOARD <FD10>  
 9ACB RETURN?  
 9ACD NO, EXIT >>9AEB  
 9ACF YES, SAVE REGISTERS <9F62>  
 9AD2 STORE IT IN LINE BUFFER (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

9A8D \*\*\*\*\* SAVE ACTUAL IN/OUT VECTORS \*\*\*\*\*  
 ---  
 9A8D COPY KSWL/H TO VECIN  
 9A98 AND CSWL/H TO VECOUT  
 9A9A IN BI GLOBAL PAGE (BE31)

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>  
 9B18 ERROR? >>9B27

9A8E 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>  
 9B18 ERROR? >>9B27

BASIC Interpreter (BI) -- V1.1.1 -- 18 JUN 84 NEXT OBJECT ADDR: 9B1A  
 -----  
 ADDR DESCRIPTION/CONTENTS  
 -----

```

9B1A PUT FILE LEVEL BACK TO ZERO
9B22 NOW FLUSH ALL OPEN FILES
9B24 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>
9B94 AND PASS CALLER'S AREG TO REMOVE CURSOR (BE3E)

```

BASIC Interpreter (BI) -- V1.1.1 -- 18 JUN 84 NEXT OBJECT ADDR: 9B97  
 -----  
 ADDR DESCRIPTION/CONTENTS  
 -----

```

9B97 RESTORE Y-REGISTER (BE40)
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 >>9BF2
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? >>9BFA
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

***** EXEC ERROR RECOVERY *****

```

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

ADDR DESCRIPTION/CONTENTS

```

9BFA CLOSE EXEC FILE <B245>
9BFD WAS ERROR "END OF DATA"?
9BFF NO, REAL ERROR THEN >>9C13
9C01 ELSE, OK -- JUST STOP EXECING
9C03 GET CURSOR HORIZONTAL POSITION
9C05 IF IN MID LINE, PASS SCREEN CHAR BACK >>9C0E
9C07 ELSE, CHANGE PROMPT TO "J"
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 "J"....
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)
9C63 AND RETURN THAT TO CALLER (0200)
9C66 RETURN

```

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

ADDR DESCRIPTION/CONTENTS

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

```

9C67 REMOVE CURSOR FROM SCREEN (BE3E)
9C6C ---
9C6E MLI: READ <BE70>
9C71 ERROR? >>9C66
9C73 GET LENGTH ACTUALLY TRANSMITTED (BEDB)
9C76 NOTHING? >>9C8E
9C79 GOT SOMETHING, FIND END OF DATA (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? (BE C8)
9CB0 NO, CONTINUE >>9CDE
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
9CDD AND EXIT TO CALLER
9CDE RETURN
9CDF GET CAT FLAG (BE4F)
9CE2 IF ZERO, GO PROCESS INDIVIDUAL ENTRIES >>9D22
9CE4 IF MINUS, GO DO SUMMARY LINE OR EXIT >>9CF9
9CE6 POSITIVE, ASSUME NULL LINE WANTED
9CE8 DROP CAT FLAG BY ONE (BE4F)

```

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

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

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

9CED ELSE, BLANK OUT \$200 AND <A66C>

9CF2 UNPACK "NAME TYPE BLOCKS ETC... <9FB0>

9CF5 LINE LENGTH IS 80

9CF7 GO RETURN IT TO CALLER >>9CD3

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

9CF9 DO SUMMARY LINE?

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 <B1D1>

9D53 NO ERROR? >>9D61

9D55 ERROR, IF RANGE ERROR...

9D57 NO, TRUE ERROR >>9D1F

9D59 RANGE ERROR, READY FOR SUMMARY LINE NEXT (BE4F)

9D5E RETURN A BLANK LINE THIS TIME >>9CD3

9D61 FORMAT FILE/DIR ENTRY INTO \$201 <A4C4>

9D64 AND RETURN IT TO CALLER >>9CF5

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

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

9D69 PROMPT = "]"?

9D6B 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)

9DDC AND EXIT TO CALLER >>9DE3

9DDE OOPS! LINE TOO LONG

9DE0 "SYNTAX ERROR" >>9AF0

9DE3 ELSE, RESTORE X REG AND EXIT (BE3F)



BASIC Interpreter (BI) -- V1.1.1 -- 18 JUN 84 NEXT OBJECT ADDR: 9DE6

BASIC Interpreter (BI) -- V1.1.1 -- 18 JUN 84 NEXT OBJECT ADDR: 9E58

ADDR DESCRIPTION/CONTENTS

ADDR DESCRIPTION/CONTENTS

```

9DE6 RETURN
9DE7 ---
9DE8 NULL LINE? >>9DF6
9DE9 NO, PUT BACK TRUE CSWL/KSWL <9A00>
9DEE SYNTAX SCAN CMD LINE <A677>
9DF1 ERROR? >>9DE0
9DF3 NO, PUT BACK BI'S INTERCEPTS <9A8D>
9DF6 ---
9DF8 MODE = 4 NOW <9F76>
9DFB RESTORE REGS AND EXIT >>9F6C
9DFE ***** WRITE BUFFERED CHARACTER *****
9DFE SAVE Y REG (BE40)
9E01 CHECK PROMPT
9E03 CHECK TO SEE IF WE ARE IN "IF", >>9E11
9E06 "PRINT", "LIST", OR "CALL" STATEMENTS >>9E11
9E09 OF AN APPLESOFT PROGRAM >>9E11
9E0B IF NOT, EXIT TO CALLER... (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 "J"?
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 ---
9E6D 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 "J"?
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 ---
9EA3 CHAR IS A RETURN?
9EA5 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
9EC1 DOES BI KNOW WE ARE TRACING? (BE41)
9EC4 YES, REAL LIVE TRACE THEN >>9F39

```

BASIC Interpreter (BI) -- V1.1.1 -- 18 JUN 84 NEXT OBJECT ADDR: 9EC4  
 ADDR DESCRIPTION/CONTENTS

9EC6 ELSE, PICK UP NEXT TOKEN ON LINE  
 9ECA IS IT A TOKEN? >>9EF1  
 9ECC OR END OF LINE? >>9EEE  
 9ECE NEITHER, DECREMENT STRING SPACE CTR (BE49)  
 9ED1 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 ITS NOT ONE BI IS INTERESTED IN >>9EEE  
 9EF9 IT IS INTERESTING, CHANGE BRANCH (9EFD)  
 9EFC AND JUMP TO ONE OF THE FOLLOWING: >>9EFE  
 9EFE IF OR PRINT: PROMPT = 0  
 9EF0 CLEAR OUT LAST CHAR SAVEAREA (BE4C)  
 9EF3 GO TO MODE = C NEXT TIME THRU (B803)  
 9EF6 (BEGIN LOOKING FOR COMMANDS) (BE38)  
 9EF7 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) -- V1.1.1 -- 18 JUN 84 NEXT OBJECT ADDR: 9F2F  
 ADDR DESCRIPTION/CONTENTS

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

\*\*\*\*\* 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 >>9EC6

9F54 LOOKING FOR A LOWER CASE "c"

9F58 LOOKING FOR A "#"

9F5A STORE CHAR TO SEARCH FOR (9F61)

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 \*\*\*\*\* PRINTER: 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) -- V1.1.1 -- 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

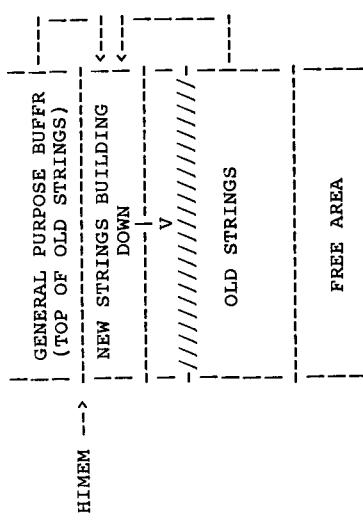
9FEE \*\*\*\*\* WRITE ONE BUFFERED BYTE \*\*\*\*\*  
 9FEE SET UP COUNT OF 0001  
 9FF2 AND JUMP INTO ROUTINE BELOW >>A007  
 9FF4 \*\*\*\*\* WRITE BUFFERED DATA/TEST ERROR \*\*\*\*\*  
 9FF4 WRITE BUFFERED DATA <A000>  
 9FF7 OK? THEN EXIT >>A01C  
 9FFA ERROR, POP OUT OF THIS SUBROUTINE  
 9FFD AND GO TO ERROR HANDLER >>9AF0

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

A000 \*\*\*\*\* WRITE ALL BUFFERED DATA \*\*\*\*\*  
 A000 ---  
 A002 GET BUFFERED DATA COUNT (BE4A)  
 A005 NONE BUFFERED? >>A01B  
 A007 STORE BUFFERED DATA COUNT IN RW PARS (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) -- V1.1.1 -- 18 JUN 84 NEXT OBJECT ADDR: A044  
 ADDR DESCRIPTION/CONTENTS

BASIC Interpreter (BI) -- V1.1.1 -- 18 JUN 84 NEXT OBJECT ADDR: A0F6  
 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  
 A05D AT LEAST 7?  
 A05B 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

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)  
 A11E NO >>A0F8  
 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) -- V1.1.1 -- 18 JUN 84 NEXT OBJECT ADDR: A16A  
 ADDR DESCRIPTION/CONTENTS

BASIC Interpreter (BI) -- V1.1.1 -- 18 JUN 84 NEXT OBJECT ADDR: A1F5  
 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

\*\*\*\*\* 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

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

A24C \*\*\*\*\* FREE BUFFER \*\*\*\*\*  
 A24C GARBAGE COLLECT STRINGS <A044>  
 A24F ERROR? >>A299  
 A255 PUT HIMEM-\$100 INTO \$3A/3B  
 A259 AND HIMEM+\$400 INTO \$3C/3D  
 A25F (COPY 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 (BE99)  
 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

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

A24C \*\*\*\*\* FREE BUFFER \*\*\*\*\*  
 A24C GARBAGE COLLECT STRINGS <A044>  
 A24F ERROR? >>A299  
 A255 PUT HIMEM-\$100 INTO \$3A/3B  
 A259 AND HIMEM+\$400 INTO \$3C/3D  
 A25F (COPY 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 (BE99)  
 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

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

A24C \*\*\*\*\* FREE BUFFER \*\*\*\*\*  
 A24C GARBAGE COLLECT STRINGS <A044>  
 A24F ERROR? >>A299  
 A255 PUT HIMEM-\$100 INTO \$3A/3B  
 A259 AND HIMEM+\$400 INTO \$3C/3D  
 A25F (COPY 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 (BE99)  
 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) -- V1.1.1 -- 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.

A2B6 ALLOCATE A BUFFER OF ANY SIZE (A=PAGES) <A1F>  
 ERROR? >>A300

A2B7 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) -- V1.1.1 -- 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 \*\*\*

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

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

A37F \*\*\*\*\* COPY BLOCK UP IN MEMORY \*\*\*\*\*  
 A37F PARTIAL PAGE? (BC92)  
 A382 NO, JUST COPY FULL PAGES NOW >>A38B  
 A384 YES, COPY SHORT PAGE FIRST <A396>  
 A387 DROP BOTH MSB'S  
 A38B PAGE COUNT GONE TO ZERO? (BC93)  
 A38E YES, DONE >>A39E  
 A390 ELSE, DROP PAGE COUNT (BC93)  
 A393 AND GO COPY A FULL PAGE UP >>A384

---  
 A396 COPY REMAINDER OF PAGE UP (BACKWARDS)  
 A397  
 A39E RETURN

BASIC Interpreter (BI) -- V1.1.1 -- 18 JUN 84 NEXT OBJECT ADDR: A39E  
 ADDR DESCRIPTION/CONTENTS

BASIC Interpreter (BI) -- V1.1.1 -- 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 \*\*\*\*\*  
 A3F9 GET STRING LENGTH  
 A3FB IGNORE NULL STRINGS >>A40C  
 A3FD POINT TO MSB OF ADDRESS  
 A3FF IS STRING STORED OUTSIDE OF PROGRAM?  
 A403 NO, LEAVE IT ALONE >>A40C  
 A405 STORE ABOVE LOMEM, ADD FACTOR TO MSB  
 A40C THEN EXIT

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  
 PAGE ALIGNEMENT)  
 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) -- V1.1.1 -- 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
A4AB 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 \*\*\*\*\* FORMAT CATALOG ENTRY LINE \*\*\*\*\*

```

A4C4 PUT OUT A BLANK LINE <A66C>
A4C7 DOUBLE QUOTE TO $200
A4CC GET LENGTH OF NAME (0259)
A4D2 COPY NAME TO LINE (0259)
A4DD ZERO ACCUMULATOR FOR LATER (BCB1)
A4E0 GET FILE TYPE (0269)
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) -- V1.1.1 -- 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 (0270)
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 (025A)
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 (025C)
A5B9 AND MINUTES (025B)
A5BE MINUTES > 60?
A5C0 NO... >>A5C3
A5C2 YES, USE ZERO MINUTES
A5C3 CONVERT MINUTES (LEFT ZERO FILL) <A60A>
A5C8 THEN PRINT A ":" (0201)
A5CC GET HOUR AGAIN
A5CF GREATER THAN 24 HOURS?
A5D1 NOPE >>A5D4
A5D3 YES, USE ZERO
    
```



BASIC Interpreter (BI) -- V1.1.1 -- 18 JUN 84 NEXT OBJECT ADDR: A5D4  
 -----  
 ADDR DESCRIPTION/CONTENTS  
 -----

A5D4 10 OR MORE HOURS (TWO DIGITS?)  
 A5D7 IN ANY CASE, CONVERT HOURS <A62F>  
 A5DB IF TWO DIGITS... >>A5DE  
 A5DD IF ONE, ADJUST LINE PTR  
 A5DE ---  
 A5E2 CONVERT YEAR (LEFT ZERO FILL) <A60A>  
 A5E6 GET MONTH INDEX (\*3) (BCB3)  
 A5E9 POINT TO LAST CHARACTER  
 A5EC COPY MONTH NAME FROM TABLE (B9BD)  
 A5EF TO LINE (0201)  
 A5F7 BACKWARDS... >>A5EC  
 A5FB PUT A "-" IN (0201)  
 A5FE TWO PLACES (0205)  
 A607 EXIT BY CONVERTING DAY >>A62F

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

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

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

A612 ---  
 A613 ISOLATE LOW NIBBLE  
 A615 AND GO CONVERT IT FIRST <A61D>  
 A619 NOW ISOLATE HIGH NIBBLE  
 A61C AND FALL THRU TO CONVERT IT ALSO  
 A61D CONVERT NIBBLE TO NUMERIC ASCII  
 A61F >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) -- V1.1.1 -- 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 <AA1B>  
 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 <AAE1>  
 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)  
 A6CB GET LENGTH OF COMMAND NAME (BE52)  
 A6D0 ALLOW 2 MORE CHARACTERS FOR NOW (BCAA)  
 A6D3 ARE ANY PARAMETERS PERMITTED? (BE54)  
 A6D6 NO...MUST BE MON OR NOMON >>A736  
 A6D8 YES, IN# OR PR#?  
 A6D9 YES... >>A739  
 A6DB ELSE, REPARSE THE COMMAND <AA1B>  
 A6E0 FOR THIS COMMAND... (BE54)

BASIC Interpreter (BI) -- V1.1.1 -- 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	---
A6EC	END OF LINE? >>A736
A6EE	NO, COMMA?
A6F0	NO >>A6F5
A6F2	YES, NO FILENAME, LOOK FOR KEYWORDS >>A787
A6F5	"/"?
A6F7	YES >>A6FD
A6F9	NO, ALPHABETIC?
A6FB	NO...FILE NAMES MUST BEGIN THAT WAY >>A72F
A6FD	---
A6FE	DON'T FLUSH ANY BLANKS OUT OF PATHNAME
A703	ALLOW 64 CHARACTERS NEXT PARSE
A709	PARSE NEXT OPERAND ON LINE <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, BAD1 >>A72C
A724	RETURN? >>A798
A726	NO, PATHNAME EXPECTED NOW? (BE54)
A72A	YES, ALL IS WELL >>A762
A72C	NO, "SYNTAX ERROR" >>A839
A72F	NON ALPHA FILE NAME, CHECK COMMAND NUMBER (BE53)
A732	IS IT "RUN"
A734	NO, ERROR >>A72C
A736	YES, ITS OK THEN (MIGHT BE "RUN 100") >>A798
A739	IN#S/PR#S, REPARSE COMMAND <AA1B>
A73C	RETURN FOUND - ERROR >>A72C
A73E	"A"? (ADDRESS KEYWORD)
A740	IF SO, GO PARSE THAT KEYWORD ONLY >>A78C
A742	ELSE, ZERO ACCUMULATOR <AB37>
A745	CONVERTING ONE BYTE'S WORTH (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) -- V1.1.1 -- 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 >>A7F8
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)
A7F8	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) -- V1.1.1 -- 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) -- V1.1.1 -- 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 <A837>
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 GET SIZE-1 IN BYTES OF VALUE (B97F)

```

BASIC Interpreter (BI) -- V1.1.1 -- 18 JUN 84 NEXT OBJECT ADDR: A954  
 ADDR DESCRIPTION/CONTENTS

A954 AND OFFSET TO VALUE IN STORAGE AREA (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 PARAM 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 \*\*\*\*\*  
 ---  
 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) -- V1.1.1 -- 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 (BB6A)  
 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) -- V1.1.1 -- 18 JUN 84 NEXT OBJECT ADDR: AA39  
 ADDR DESCRIPTION/CONTENTS

BASIC Interpreter (BI) -- V1.1.1 -- 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  
 AA3A IGNORE BLANKS  
 AA3F GET NEXT NON-BLANK <AA4A>  
 AA42 COMMA?  
 AA44 YES, OUT >>AA49  
 AA46 RETURN?  
 AA48 EXIT INDICATING WHAT WE FOUND  
 AA49 RETURN  
 AA4A \*\*\*\*\* GET NEXT CHARACTER \*\*\*\*\*  
 AA4A GET NEXT CHAR IN INPUT LINE (0200)  
 AA4D FORCE OFF MSB  
 AA4F LOWER CASE?  
 AA51 NO >>AA55  
 AA53 YES, FORCE UPPER CASE  
 AA55 BUMP LINE INDEX  
 AA56 IS THIS A FLUSH CHARACTER (LIKE BLANK)? (BCA9)  
 AA59 YES, GO GET NEXT ONE >>AA4A  
 AA5B ELSE, RETURN WITH IT  
 AA5C \*\*\*\*\* CONVERT DIGIT AND ADD TO ACCUM \*\*\*\*\*

AAAE \*\*\*\*\* CONVERT HEX DIGIT AND ADD \*\*\*\*\*  
 AAAE NUMERIC?  
 AAB0 NO >>AABE  
 AAB4 YES >>AAC4  
 AAB6 NON-NUMERIC, HOW BOUT "A" THRU  
 AABA "F"  
 AABC 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  
 AAD7 \*\*\*\*\* SHIFT 3 BYTE ACCUM LEFT A BIT \*\*\*\*\*  
 AAD7 SHIFT THE THREE BYTE WORK ACCUM (BCAF)  
 AAE0 RETURN  
 AAEE \*\*\*\*\* SCAN CMD TABLE FOR COMMAND \*\*\*\*\*

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... (BCBI)  
 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 \*\*\*\*\*  
 AAEE \*\*\*\*\* SCAN CMD TABLE FOR COMMAND \*\*\*\*\*  
 AAEE START WITH LAST COMMAND IN TABLE  
 AAEE IS IT A "-" COMMAND? (BCBD)  
 AAEB NOPE >>AAF5  
 AAED 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 (B858)  
 AAFD SAME LENGTH AS LAST NAME? >>AB05  
 AAFF 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) -- VI.1.1 -- 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 (BEB8)  
 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>  
 ABA4 RESTORE OLD HIMEM  
 ABA6 ERROR? >>AC14  
 ABAB NO, CLEAR VARIABLES <D665>  
 ABB0 THEN GO "RUN" PROGRAM >>ABC7

BASIC Interpreter (BI) -- VI.1.1 -- 18 JUN 84 NEXT OBJECT ADDR: ABB0  
 ADDR DESCRIPTION/CONTENTS

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

ABB2 NO INPUT FILE ACTIVE NOW  
 ABB7 NO APPLESOFT ERROR NUMBER  
 ABB8 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) <E273>  
 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 \*\*\*\*\* WARMDOS: 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 LOWEM = ARRAYS = FREESTART  
 AC0A ALL TO END OF PROGRAM LOADED  
 AC14 RETURN

BASIC Interpreter (BI) -- V1.1.1 -- 18 JUN 84 NEXT OBJECT ADDR: AC14  
 ADDR DESCRIPTION/CONTENTS

BASIC Interpreter (BI) -- V1.1.1 -- 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 *****
---
AC61 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 # (BEG8)
ACA9 THEN CALL APPLESOFT TO FIND THE LINE <D61A>
ACAF SUBTRACT ONE FROM THE ADDRESS
ACB1 AND POINT APPLESOFT'S GETCHR SUBROUTINE
ACB3 AT IT (SO NEXT CHAR READ WILL BE FIRST
ACB5 CHARACTER ON THE LINE).
ACBA RETURN
  
```

```

ACBB ***** "SAVE" COMMAND *****
ACBB DOES FILE EXIST ALREADY? >>ACDF
ACBF 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
ACEE 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?
AD43 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 >>ADGE
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) -- V1.1.1 -- 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  
ADAB 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 >>ADD1  
ADB4 NO, NEW LINE <9FAB>  
ADBC END OF NAME YET? >>ADC9  
ADBE NO, COPY NAME IN PATHNAME1 BUFFER (BCBD)  
ADC3 TO OUTPUT DEVICE <9FAD>  
ADC9 AND SKIP A BLANK LINE <9FAB>  
ADD0 DONE

BASIC Interpreter (BI) -- V1.1.1 -- 18 JUN 84 NEXT OBJECT ADDR: ADD0

-----  
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  
ADDA NO, NEW FILE (BE57)  
ADDD 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) -- V1.1.1 -- 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 OCCURRED >>AE92

AE74 OR ELSE, "RANGE ERROR"
AE77 RETURN

AE78 "INVALID PARM" ERROR
AE7B RETURN

AE7C ---
AE7E MLI: GET EOF <BE70>
AE81 ERROR? >>AE90
AE83 GET L (EOF MARK) (BE8C)
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 BLOOD/BRUNING)?
AEBB 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
AED0 WRITING
AED2 MLI: READ OR WRITE <BE70>
AED4 ERROR? >>AE90
AED4 THEN EXIT THRU CLOSE >>AF94
  
```

BASIC Interpreter (BI) -- V1.1.1 -- 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) -- V1.1.1 -- 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 \*\*\*\*\*  
 AF94 SET MLI CLOSE OPCODE  
 AF96 AND GO TO MLI >>AFA4

AF98 \*\*\*\*\* READ/WRITE A RANGE \*\*\*\*\*  
 AF98 READ MLI OPCODE  
 AF9A JUMP IN >>AF9E  
 AF9C WRITE MLI OPCODE  
 AF9E STORE LENGTH (BEDA)  
 AFA4 EXIT THRU MLI:READ OR WRITE >>BE70

AFA7 \*\*\*\*\* "PR#" COMMAND \*\*\*\*\*  
 AFA7 USE CSWL AND OUTVEC  
 AFA8 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  
 AFB8 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)  
 AFD8 AND REPLACE ONE OR THE OTHER WITH (0036)  
 AFE0 HIS ADDRESS (BE59)

BASIC Interpreter (BI) -- V1.1.1 -- 18 JUN 84 NEXT OBJECT ADDR: AFE6

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

AFE6 RETURN  
 AFE7 VALIDITY CHECK AD KEYWORD VALUE <AFF9>  
 AFEA 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) -- V1.1.1 -- 18 JUN 84 NEXT OBJECT ADDR: B054

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 <B0D1>  
 B07B ERROR? >>B0B7  
 B07D GET TYPE REQUESTED FOR SEARCH (BE6A)  
 B080 ANY TYPE WILL DO? >>B087  
 B082 NO, CHECK TYPE AGAINST THIS ENTRY (0269)  
 B085 NOT IT, SKIP IT >>B08D  
 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

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) -- V1.1.1 -- 18 JUN 84 NEXT OBJECT ADDR: B0D7

ADDR DESCRIPTION/CONTENTS

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) (BE08)  
 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 (BEEC)  
 B13E ... - BLOCKS\_USED (BEBD)  
 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

BASIC Interpreter (BI) -- V1.1.1 -- 18 JUN 84 NEXT OBJECT ADDR: B193  
ADDR DESCRIPTION/CONTENTS

B194 \*\*\*\*\* OPEN FILE \*\*\*\*\*  
A REGISTER = ACCESS BITS  
X REGISTER = DEFAULT TYPE

B198 T KEYWORD GIVEN?  
B199 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

B1C9 "FILE TYPE MISMATCH"  
B1CC RETURN  
B1CD "FILE LOCKED"  
B1D0 RETURN

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 0 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) -- V1.1.1 -- 18 JUN 84 NEXT OBJECT ADDR: B21D  
ADDR DESCRIPTION/CONTENTS

B21E \*\*\*\*\* EXTERNAL COMMAND HANDLER \*\*\*\*\*

B21E INDIRECT JMP TO XTRNAD VECTOR >>BE50

B221 \*\*\*\*\* "EXEC" COMMAND \*\*\*\*\*

B221 IS THIS FILE OPEN ALREADY? <B41F>  
B224 NO >>B250  
B226 YES, EXEC CLOSING? (BE4E)  
B229 NO >>B24C  
B22B SAVE REFNUM (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 ARG  
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

BASIC Interpreter (BI) -- V1.1.1 -- 18 JUN 84 NEXT OBJECT ADDR: B278

-----  
 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

BASIC Interpreter (BI) -- V1.1.1 -- 18 JUN 84 NEXT OBJECT ADDR: B2FA

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

B2FB \*\*\*\*\* 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

B30C \*\*\*\*\* "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

B31F \*\*\*\*\* FLUSH ALL OPEN FILES \*\*\*\*\*

B31F REFNUM = 0 (ALL FILES)  
 B321 JUMP INTO FLUSH >>B32F

B323 \*\*\*\*\* "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

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

B338 ---  
 B339 LOOK UP NAME IN OPEN FILE LIST <B41F>  
 B33C NOT CURRENTLY OPEN? >>B34B  
 B33E ---  
 B33F IT IS OPEN, "FILE BUSY" ERROR  
 B342 RETURN

BASIC Interpreter (BI) -- V1.1.1 -- 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 (BEB7)  
 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 (BEB8)  
 B391 AGAINST HIS "t" VALUE (BE6A)  
 B394 MISMATCH? >>B343  
 B396 NO, TYPE = TXT?  
 B398 NO >>B3AD  
 B39A YES, GET RECORD LENGTH FROM AUXID (BEEA)  
 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 <ALF5>  
 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 (BEB8)  
 B3CE "DIR" FILE?  
 B3D0 YES >>B3D3  
 B3D2 NO  
 B3D3 ---

BASIC Interpreter (BI) -- V1.1.1 -- 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/RECLEN 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 (BB4E)  
 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) -- V1.1.1 -- 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 >>B498  
 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 ----  
 B49C 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 >>B4FE  
 B4E6 YES, EXEC NO LONGER ACTIVE (BE43)  
 B4E9 AND NO LONGER CLOSING (BE4E)  
 B4ED RETURN TO CALLER

BASIC Interpreter (BI) -- V1.1.1 -- 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 ----  
 B505 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  
 B522 "F" OR "R" GIVEN? (BE57)  
 B527 NO, INVALID PARM >>B57D  
 B529 BOTH GIVEN?  
 B52B YES, INVALID PARM >>B57D  
 B52D JUST "R" GIVEN?  
 B52F NO, JUST "F" >>B53D  
 B531 JUST "R", COPY "R" VALUE TO "F" (BE65)  
 B534 ("R" AND "F" ARE 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 \$8D (BED3)  
 B556 MLI: SET NEWLINE <BE70>  
 B559 ERROR? >>B57F  
 \*\*\*\*\* SKIP LINES BY READING THEM \*\*\*\*\*  
 B55B ----  
 B55E "F" = 0? (BE64)  
 B562 YES, DONE >>B580  
 B564 ELSE...  
 B566 MLI: READ NEXT FIELD (LINE) <BE70>  
 B569 ERROR? >>B57F  
 B56E DECREMENT "F" VALUE BY ONE

BASIC Interpreter (BI) -- V1.1.1 -- 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" \* RECLN \*\*\*\*\*  
 B59E SHIFT "R" VALUE RIGHT (BEG6)  
 B5A6 IF LOW BIT OFF, NO ADD >>B5BF  
 B5A9 ADD ONE INSTANCE OF RECLN TO MARK (BCAF)  
 B5B8 OVERFLOW? >>B5D2  
 B5BD ACCUM OVERFLOW? >>B5D2  
 B5BF SCALE ACCUM (MULTIPLIER) UP BY 2 (BCAF)  
 B5C8 IF "R" NON ZERO... (BEG5)  
 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/\$8D (OR NONE IF "L") (BED3)  
 B621 MLI: SET NEWLINE <BE70>  
 B624 ERROR? >>B62B  
 ----

BASIC Interpreter (BI) -- V1.1.1 -- 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)  
 B62F AND GET/SET LIST REFNUM (BEC7)  
 B634 READING TO \$259 (BED7)  
 B63E INIT CAT FLAG TO FIRST LINE VALUE (BE4F)  
 B644 "R" GIVEN?  
 B647 NO, DONE >>B626  
 B64B YES, ZERO OUT MARK (BEC8)  
 B656 MLI: REWIND FILE <BE70>  
 B659 ERROR? >>B660  
 B65D MARK INPUT FILE ACTIVE (BE44)  
 B660 AND EXIT

B661 \*\*\*\*\* "RANGE ERROR" \*\*\*\*\*  
 B661 "RANGE ERROR" CODE  
 B665 EXIT TO CALLER

B666 \*\*\*\*\* PRE-POSITION FOR I/O \*\*\*\*\*  
 B666 ----  
 B669 "B", "F", OR "R" GIVEN?  
 B66B NO, EXIT >>B6AF  
 B66D "R"?  
 B66F NO >>B67B  
 B671 YES, COMPUTE ABSOLUTE POSITION <B581>  
 B674 ERROR? >>B661  
 B676 NO, SET MARK TO NEW POSITION <B6A8>  
 B679 ERROR? >>B6B0  
 B67B "F" GIVEN? (BE57)  
 B680 NO >>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 (BESA)  
 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) -- V1.1.1 -- 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 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  
 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) -- V1.1.1 -- 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 REFNUM\*32 FOR INDEX INTO  
 B739 FILE NAME TABLE  
 B73E SAVE CURRENT "L" VALUE IN OPEN FILE (BCFF)  
 B741 NAME TABLE AND IN CURRENT RECLEN (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  
 B768 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) -- V1.1.1 -- 18 JUN 84 NEXT OBJECT ADDR: B7D7  
-----  
ADDR DESCRIPTION/CONTENTS  
-----

B7D9 \*\*\*\*\* SET FILE INFO \*\*\*\*\*  
B7D9 MODIFIED TIME/DATE = 0  
B7E7 SET NUMBER OF PARAMS (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

B805 \*\*\*\*\* SYSCTBL \*\*\*\*\*  
LSB'S OF MLI CALL PARAMETER LISTS IN THE  
BI GLOBAL PAGE (\$BEXX)  
  
B805 CREATE: \$A0 DESTROY: \$AC RENAME: \$AF  
B808 SFI: \$B4 GFI: \$B4 ONLINE: \$C6  
B80B SFFX: \$AC GPF: \$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  
  
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) -- V1.1.1 -- 18 JUN 84 NEXT OBJECT ADDR: B859  
-----  
ADDR DESCRIPTION/CONTENTS  
-----

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 'BSAVERIFYBLOADDELETEBYECATALOGOPE'  
B898 'NWRITECREATEFRESTORENAMEBRUNLO'  
B8B8 'CKCHAIN#FLUSHREADPOSITIONOMONPR#'  
B8D8 'PREFIXCLOSEAPPEND'

B8E9 \*\*\*\*\* COMMAND HANDLER ADDRESS TABLE \*\*\*\*\*  
ADDRESSES OF THE COMMAND HANDLER ROUTINES  
FOR EACH COMMAND IN THE ORDER GIVEN ABOVE.

(EXTERNAL)

B8E9 B8EB B8ED B8EF B8F1 B8F3 B8F5 B8F7 B8F9 B8FB B8FD B8FF B901 B903 B905 B907 B909 B90B B90D B90F B911 B913 B915 B917 B919 B91B B91D  
EXEC PR# CAT FRE BYE RUN BRUN EXEC LOAD LOCK OPEN READ SAVE BLOAD BSAVE CHAIN CLOSE FLUSH NOMON STORE WRITE APPEND CREATE DELETE PREFIX RENAME

BASIC Interpreter (BI) -- V1.1.1 -- 18 JUN 84 NEXT OBJECT ADDR: B91F

ADDR	DESCRIPTION/CONTENTS
B91F	UNLOCK
B921	VERIFY
B923	CATALOG
B925	RESTORE
B927	POSITION
B929	"-" COMMAND

B92B \*\*\*\*\* PERMITTED KEYWORDS FOR CMDS \*\*\*\*\*  
 TWO BYTES PER COMMAND IN THE ORDER ABOVE.  
 EACH ENTRY HAS 16 BIT SETTINGS FOR THE  
 PARAMETERS PERMITTED ON THAT COMMAND.  
 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 = "@" (LINE NO.) PERMITTED  
 0004 = "S" AND/OR "D" (SLOT/DRIVE)  
 0002 = "F" (FIELD) PERMITTED  
 0001 = "R" (RECORD) PERMITTED  
 ("V" IS IGNORED)

C	P	S	D	E	N	T	P	A	B	E	L	@	S	F	R
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	.	.	.	.	.	.	.	.	.	.	.	.	.	.

BASIC Interpreter (BI) -- V1.1.1 -- 18 JUN 84 NEXT OBJECT ADDR: B949

ADDR	DESCRIPTION/CONTENTS
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	"_"

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 ---

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
B988	@:	2	BYTES	AT	+11

B989 \*\*\*\*\* FILE TYPES TABLES \*\*\*\*\*  
 FILE TYPE CODES, GIVEN IN INVERSE ORDER  
 TO FILE TYPE NAMES WHICH FOLLOW.

BASIC Interpreter (BI) -- V1.1.1 -- 18 JUN 84 NEXT OBJECT ADDR: B989  
 -----  
 ADDR DESCRIPTION/CONTENTS  
 -----

B989 \$FF = "SYS"  
 B98A \$FE = "REL"  
 B98B \$FD = "VAR"  
 B98C \$FC = "BAS"  
 B98D \$FB = "IVR"  
 B98E \$FA = "INT"  
 B98F \$F9 = "CMD"  
 B990 \$F8 = "DIR"  
 B991 \$06 = "BIN"  
 B992 \$04 = "TXT"  
 B993 \$EF = "PAS"  
 B994 \$1A = "AWP"  
 B995 \$1B = "ASP"  
 B996 \$19 = "ADB"  
 B997 ---  
 B997 'ADBSPAWPPASTBTBINDIRCMDINTIVRBASVARRELSYS'  
 B9C1 \*\*\*\*\* MONTH TABLE \*\*\*\*\*  
 B9C1 'JANFEBMARAPRMAXJUNJULAUUGSEPNOVDEC'  
 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 \*\*\*\*\* INDEXS TO PACKED MESSAGES \*\*\*\*\*  
 BY BI ERROR NUMBER  
 BA15 ---  
 BA29 \*\*\*\*\* COMMON LETTERS IN MESSAGES \*\*\*\*\*  
 BA29 'ACDEFILMNORTU '

BASIC Interpreter (BI) -- V1.1.1 -- 18 JUN 84 NEXT OBJECT ADDR: BA38  
 -----  
 ADDR DESCRIPTION/CONTENTS  
 -----

BA38 \*\*\*\*\* LESS COMMON LETTERS \*\*\*\*\*  
 BA38 ---  
 BA39 'BGHKFSVWXY/(:) :'  
 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  
 BAC0  
 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) -- V1.1.1 -- 18 JUN 84 NEXT OBJECT ADDR: BC92  
 -----  
 ADDR DESCRIPTION/CONTENTS  
 -----

BASIC Interpreter (BI) -- V1.1.1 -- 18 JUN 84 NEXT OBJECT ADDR: BAF8  
 -----  
 ADDR DESCRIPTION/CONTENTS  
 -----

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  
 BCAB 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  
 BCBA MONTH  
 BCBB DAY  
 BCBB YEAR  
 BCBB ERROR MSG LEN OR LINE LEN FOR CAT/CATALOG  
 BCBB ENTRY LENGTH IN DIRECTORY FILE  
 BCBB ENTRIES PER BLOCK IN DIRECTORY FILE  
 BCBB FILE COUNT FROM DIRECTORY FILE  
 BCBB DIRECTORY ENTRY NUMBER COUNTER

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  
 BB47 \*\*\*\*\* VARIABLES \*\*\*\*\*  
 BB47 NUMBER OF PAGES TO ALLOCATE/FREE  
 BB48 NOT USED  
 BB49 TOP OF BUFFERS FOR GARBAGE COLLECTION  
 BB4A BOTTOM OF BUFFERS

BCBC \*\*\*\*\* PATHNAME 1 BUFFER \*\*\*\*\*  
 BCBC COMMAND OR PATH LENGTH  
 BCBD TXBUF (COMMAND OR PATHNAME STRING)  
 BCBD NOT USED

BB4B \*\*\*\*\* \$BB4B-\$BC7A NOT USED \*\*\*\*\*  
 BB4B NOT USED  
 BC7B \*\*\*\*\* VARIABLES \*\*\*\*\*

BCFE \*\*\*\*\* OPEN FILE NAME TABLE \*\*\*\*\*  
 (EACH ENTRY IS 32 BYTES LONG)  
 (THERE ARE 8 ENTRIES)  
 BCFF FILE 0: LENGTH OF NAME  
 BD00 FILE 0: L VALUE LSB  
 BD01 FILE 0: L VALUE MSB  
 (FILE NAME IS STORED BACKWARDS)  
 BDFE LAST 2 BYTES NOT USED

BC7B SAVED HIMEM VALUE DURING CHAIN LOAD  
 \*\*\*\*\* GARBAGE COLLECT MARKED GC: \*\*\*\*  
 GC: HIRANGE - WORKSAREASIZE  
 GC: WORKAREA MSB  
 GC: NUMBER OF PAGES IN WORKAREA  
 GC: LORANGE (START OF STRINGS TO COPY)  
 GC: HIRANGE (END OF STRINGS TO COPY)  
 ARRAYS START LSB  
 ARRAYS ENDING MSB+1  
 GC: START OF STRING AREA (ALSO PGM START)  
 GC: END OF STRING AREA  
 MSB ADJUST FACTOR FOR STRING POINTERS  
 PAGE FOLLOWING BLOCK BUFFER  
 \*\*\*\*\* STORED VARIABLES FILE HEADER \*\*\*  
 COMBINED LEN OF SIMPLE/ARRAY VARS  
 LEN OF SIMPLE VARS ONLY  
 HIMEM WHEN VARS WERE COMBINED  
 \*\*\*\*\*  
 POINTER TO COMBINED VARIABLES/STRINGS  
 LENGTH OF COMBINED VARIABLES/STRINGS  
 LENGTH OF STRINGS ONLY

### 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	VDSYSIO	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	DIRFLG	PREFIX read active flag (MSB on).	
BE48	STRINGS	File being READ is a DIR file (MSB on).	
BE49	TBUFPTTR	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.	
BE4E	YXFILE	Previous output character (for recursion check).	
BE4F	CATFLAG	Number of files open (not counting EXEC).	
BE50-BE51	XTRNADDR	EXEC file being closed flag (MSB on).	
BE52	XLEN	Line type to format next in DIR file READ.	
		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    \$0A = OPEN \$01 = IN#        \$0B = READ \$02 = PR#        \$0C = SAVE \$03 = CAT        \$0D = BLOAD \$04 = FRE        \$0E = BSAVE \$05 = RUN        \$0F = CHAIN \$06 = BRUN       \$10 = CLOSE \$07 = EXEC       \$11 = FLUSH \$08 = LOAD       \$12 = NOMON \$09 = SAVE       \$13 = STORE  \$14 = WRITE \$15 = APPEND \$16 = CREATE \$17 = DELETE \$18 = PREFIX \$19 = RENAME \$1A = UNLOCK \$1B = VERIFY \$1C = CATALOG \$1D = RESTORE \$1E = POSITION

BE54-BE55	PBITS	Permitted command operands bits: \$8000 Prefix needed. Pathname optional. \$4000 Slot number only (PR# or IN#). \$1000 File name optional. \$0800 If file does not exist, create it. \$0400 T: file type permitted. \$0200 Second file name required. \$0100 First file name required. \$0080 AD: address keyword permitted. \$0040 B: byte offset permitted. \$0020 E: ending address permitted. \$0010 L: length permitted. \$0008 @: line number permitted. \$0004 S or D: slot/drive permitted. \$0002 F: field permitted. \$0001 R: record permitted. (V always permitted but ignored.)
-----------	-------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

BE56-BE57	FBITS	Operands found on command line. Same bit assignments as above.
BE58-BE59	VADDR	A keyword value.
BE5A-BE5C	VBYTE	B keyword value.
BE5D-BE5E	VENDA	E keyword value.
BE5F-BE60	VLNTH	L keyword value.
BE61	VSLOT	S keyword value.
BE62	VDRIV	D keyword value.
BE63-BE64	VFELD	F keyword value.
BE65-BE66	VRECD	R keyword value.
BE67	VVOLM	V keyword value (ignored).
BE68-BE69	VLINE	@ keyword value.
BE6A	VTYPE	T keyword value (in hex).
BE6B	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	SYSCALL	MLI call number for this call.
BE86-BE87	SYSPARM	Address of MLI parameter list for this call.
BE88-BE8A	BADCALL	Return from MLI call.
BE8B-BE9E		MLI error return: translate error code to BI error number.
BE9F	BISPAREL	Not used.
BEA0-BEAB	SCREATE	CREATE parameter list.
BEAC-BEAE	SSGPRFX	GET_PREFIX, SET_PREFIX, DESTROY parameter list.
BEAF-BEB3	SRENAME	RENAME parameter list.
BEB4-BEC5	SSGINFO	GET_FILE_INFO, SET_FILE_INFO parameter list.
BE6C-BECA	SONLINE	ONLINE, SET_MARK, GET_MARK, SET_EOF, GET_EOF, SET_BUF, GET_BUF, QUIT parameter list.
BE6B-BED0	SOPEN	OPEN parameter list.
BED1-BED4	SNEWLN	SET_NEWLINE parameter list.
BED5-BEDC	SREAD	READ, WRITE parameter list.
BEDD-BEDE	SCLOSE	CLOSE, FLUSH parameter list.
BEDF-BEF4	CCCSPARE	"COPYRIGHT APPLE, 1983"
BEF5-BEF7	GETBUFR	GETBUFR buffer allocation subroutine vector.
BEF8-BEFA	FREEBUFR	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
DESCRIPTION/CONTENTS

Disk Controller Boot ROM -- Apple II/II+/IIE NEXT OBJECT ADDR: C617
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 \*\*\*\*\*

SECTOR BUFFER POINTER
SLOT NUMBER \* 16 FOR INDEX
WORKBYTE
SECTOR WANTED
TRACK FOUND
TRACK WANTED

\*\*\*\*\* EXTERNAL ADDRESSES \*\*\*\*\*
SYSTEM STACK
AUXILIARY BUFFER
TRANSLATE TABLE
SECTORS TO LOAD
ENTRY POINT
PHASE0 OFF
PHASE0 ON
MOTOR ON
DRIVE SELECT
READ DATA REGISTER
SET READ MODE
MONITOR WAIT ROUTINE
RTS

C600 \*\*\*\*\* BUILD READ TRANSLATE TABLE \*\*\*\*\*
C602 SIGNATURE
C602 INITIALIZE TABLE VALUE INDICATOR
C606 STORE BIT PATTERN
C609 SHIFT PATTERN LEFT ONE BIT
C609A ARE THERE ANY TWO ADJACENT BITS ON?
C609C NO, TRY ANOTHER PATTERN ->C61E
C609E 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

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 \*\*\*\*\*
C652 ---
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 XIXLXIXI  
 C68D SAVE FOR LATER  
 C68F READ REGISTER FOR NEXT BYTE (C08C)  
 C692 LOOP UNTIL VALID >>C68F  
 C694 COMBINE WITH PREVIOUS IXLXIXIX AND XLXIXIXI  
 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

```

*****
* * * * *
* * 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. * * * * *
* * * * *
*****

```

\*\*\*\*\* 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

```

\*\*\*\*\* EXTERNAL ADDRESSES \*\*\*\*\*

```

AUXILIARY BUFFER
TRANSLATE TABLE
SCREEN LOCATION
SECTORS TO LOAD
ENTRY POINT
PHASE0 OFF
PHASE0 ON
MOTOR OFF
MOTOR ON
READ DATA REGISTER
SET READ MODE
DRIVE SELECT
MONITOR WAIT ROUTINE
FCAB

```

C552    \*\*\*\*\* SLOT5 CODE \*\*\*\*\*

THE FOLLOWING TWO ROUTINES ARE IN THE \$C500  
 AREA BUT ARE USED BY THE \$C600 LOGIC.

C552    \*\*\*\*\* BOOTFAIL \*\*\*\*\*  
 COME HERE IF BOOT FAILS. PUT MESSAGE ON  
 SCREEN AND GO TO SLEEP FOREVER.

C552    17 CHARACTERS IN MESSAGE  
 C557    PUT AT BOTTOM OF SCREEN (07DB)  
 C55D    THEN GO TO SLEEP >>C55D

C55F    'Check Disk Drive'

C56F    \*\*\*\*\* SKIP OVER MISCELLANEOUS CODE \*\*\*\*\*

C56F    SLOT 5 LOGIC IN HERE

C58E    \*\*\*\*\* BUILD READ TRANSLATE TABLE \*\*\*\*\*

```

C58E    INITIALIZE BIT PATTERN
C590    INITIALIZE TABLE VALUE INDICATOR
C592    STORE BIT PATTERN
C595    SHIFT PATTERN LEFT ONE BIT
C596    ARE THERE ANY TWO ADJACENT BITS ON?
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
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: 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  
 C656 ---  
 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 \$AA  
 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 XIXLXIX1  
 C68D SAVE FOR LATER  
 C68F READ REGISTER FOR NEXT BYTE (C08C)  
 C692 LOOP UNTIL VALID >>C68F  
 C694 COMBINE WITH PREVIOUS XIXLXIX AND XIXLXIX1  
 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: 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 ---  
 C60B INITIALIZE SLOT (6)  
 C60D INITIALIZE DEVICE (1 OR 2)  
 C60F SAVE DRIVE NUMBER ON STACK  
 C610 INSURE READ MODE (C08E)  
 C616 GET DRIVE NUMBER BACK  
 C617 SELECT APPROPRIATE DRIVE (C0EA)  
 C61A TURN MOTOR ON (C089)

C61D \*\*\*\*\* RECALIBRATE DISK ARM \*\*\*\*\*

C61D 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 \*\*\*\*\*

C634 ---  
 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: 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

```

**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          DEVLP          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 7-26**

Expand the 40-track drive patch to show how to patch PRODOS versions 1.0.2 and 1.1.1 as well as 1.0.1.

This patch modifies the Disk II Driver, which is a part of the "PRODOS" file, so that it allows 320 blocks per volume instead of 280 blocks per volume.

```
UNLOCK PRODOS
BLOAD PRODOS,TSYS,A$2000
CALL -151
address*:40
3D0G
BSAVE PRODOS,TSYS,A$2000
LOCK PRODOS
```

\*"address" varies with the version of PRODOS, as follows:

<b>PRODOS Version</b>	<b>address</b>
1.0.1	520D
1.0.2	52CD
1.1.1	56E3

---

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



## Quality Software Products For the Apple

### BOOKS

#### **Beneath Apple ProDOS** by Don Worth & Pieter Lechner

Describes the ProDOS Operating System clearly and in detail, going beyond Apple's manuals. Many programming examples are included. 288 pages. 176 pages. **\$19.95**

#### Supplements to Beneath Apple ProDOS:

Versions 1.0.1 and 1.0.2 (combined) **\$10.00**  
Version 1.1.1 **\$12.50**

#### **Beneath Apple DOS** by Don Worth & Pieter Lechner

The popular best seller that covers all facets of DOS 3.3 and previous Apple disk operating systems. 176 pages. **\$19.95**

#### **Understanding the Apple II** by Jim Sather

Foreword by Steve Wozniak. A definitive source of information, covers Apple II and Apple II Plus hardware, including the disk controller and logic state sequencer. 352 pages. **\$22.95**

#### **Understanding the Apple IIe** by Jim Sather

The companion to **Understanding the Apple II**, this book covers Apple IIe hardware, including video graphics and the 1985 firmware upgrade (65C02). 368 pages. **\$24.95**

### UTILITIES

#### **Bag of Tricks 2** by Don Worth & Pieter Lechner

Quality Software's popular set of Apple II disk utility programs, **Bag of Tricks**, has been thoroughly revised and updated for the ProDOS operating system. TRAX, INIT, ZAP, and FIXCAT are the four comprehensive utility programs, all with improved user interfaces to make them easier to use than the original **Bag of Tricks**.\* Unprotected diskette and 200-page manual. 64K. **\$49.95**

\*Special offer to **Bag of Tricks** owners--save \$20 by ordering directly from Quality Software. To order, send in your **Bag of Tricks** diskette and \$29.95, plus shipping, handling, and sales tax. We will return your diskette along with the new product.

#### **Universal File Conversion** by Gary Charpentier

Moves programs and data among the five operating systems used on the Apple II family of computers: DOS, ProDOS, CP/M, Pascal, and SOS. Unprotected diskette and 48-page manual. 64K. **\$34.95**

**Ordering directly from Quality Software**

To order our products directly, mail this order form to Quality Software (at the address below) with your payment--the price of the software (plus sales tax if shipped to California) plus shipping and handling charges. Your payment can be a check or bank draft made payable to Quality Software in US dollars, or your VISA or MASTERCARD number and expiration date (VISA and MASTERCARD holders may phone in their orders). California residents must add the appropriate sales tax (6%, 6.5%, or 7%).

**Shipping charges:**

48 Continental United States (UPS).....\$2.50  
Alaska, Hawaii, Canada, and Mexico (air mail).....\$5.00  
All other countries (insured air mail).....\$10.00

Send your order to:

**QUALITY SOFTWARE**  
21610 Lassen Street #7  
Chatsworth CA 91311  
(818) 709-1721

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 Version 1.1.1



by Don Worth and Pieter Lechner

**QS** QUALITY  
SOFTWARE