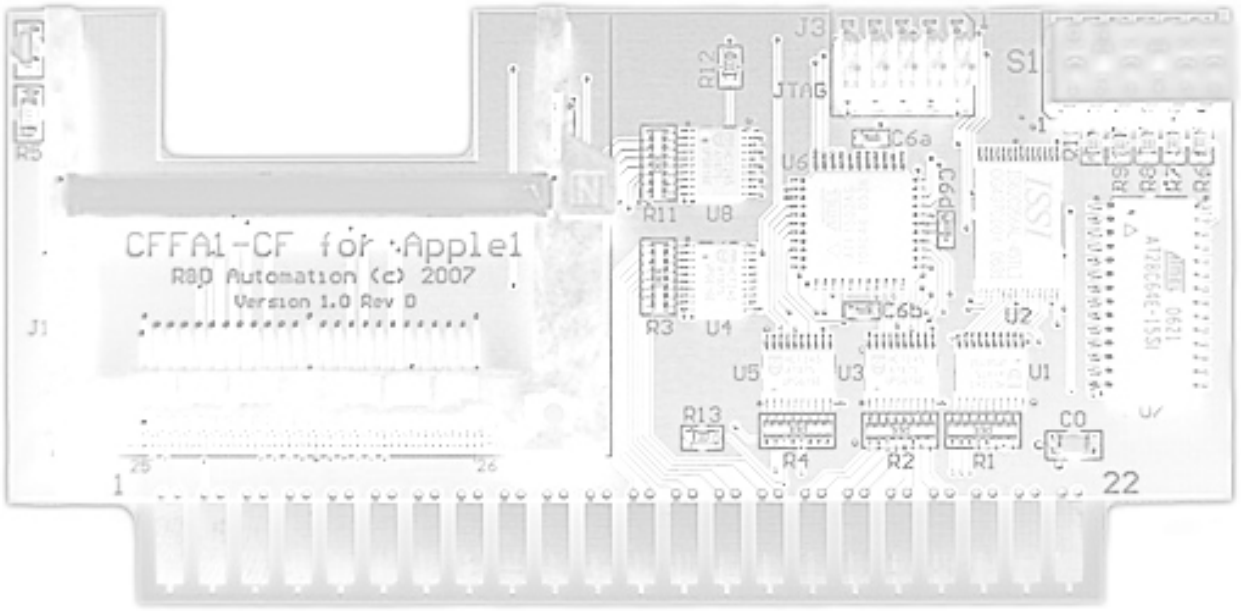
CFFA1 – CompactFlash Interface for Apple 1



Manual v1.0 for CFFA1 v1.0 RevD by Richard Dreher

Disclaimer of All Liability

Plain English Version:

Do not use this manual or the CFFA1 Interface card for any mission-critical applications, or for any purpose, in which a bug or failure could cause you a financial or material loss. This product was designed to enhance your Apple 1 computing experience, but may contain design flaws that could inhibit its proper operation, or result in a loss of the data recorded on the storage devices attached to it. When using this product you assume all risks associated with operation or data loss. If these terms are not acceptable, you may return the product for a refund.

Legalese Version:

Richard Dreher, doing business as R&D Automation, makes no warranties either express or implied with respect to this manual or with respect to the software or firmware described in this manual, its quality, performance, or fitness for any particular purpose. All software and firmware is sold or licensed “as is”. Any risk of incidental or consequential damages resulting from the use of the information in this manual or the software / firmware / hardware described herein, shall be assumed by the user or buyer or licensee. In no event will R&D Automation be liable for direct, indirect, incidental or consequential damages resulting from any defect in the software / firmware / hardware described in this manual.

R&D Automation reserves the right to make changes and improvements to the product described in this manual at any time and without notice.

Contents

Basic Information

Warranty and Return Information

You may return the CFFA1 Interface card for any reason within 30 days of receiving it. This should allow you enough time to evaluate the compatibility with your system. I guarantee your CFFA1 Interface card to be free of defects under normal usage for a period of one year from the date you receive the product. This means that if the card fails, and you have treated it properly, I will repair, replace, or refund your money at my discretion, to be determined by me on a case-by-case basis.

If you want to return the product under warranty, please contact me via E-mail to discuss return arrangements. Include your name and the serial number from the sticker on the back of the card. It is your responsibility to get the product you are returning back to my door. I will not be responsible for lost shipments. Please choose shipping methods and insurance, as you deem necessary.

Warnings

You should avoid electrostatic discharge to the CFFA1 Interface card. Like all electronics devices, static “shock” can destroy or shorten the life span of the CFFA1 Interface card. Avoid touching the CFFA1 Interface card after you have walked across the room, especially over carpet, and especially in dry weather.

You should safely discharge yourself before you handle the CFFA1 Interface card. This can be done by momentarily coming into contact with a grounded piece of metal.

In all cases, please exercise common sense and observe all electrical warnings provided by the manufacturers of the equipment you are using.

Quick Start Instructions

1. Be sure that your Apple1 has a jumper between address select line A and T, as shown in Figure 1. Note: *Replica1 computers have a static memory mapping scheme and need no modifications for the CFFA1.*

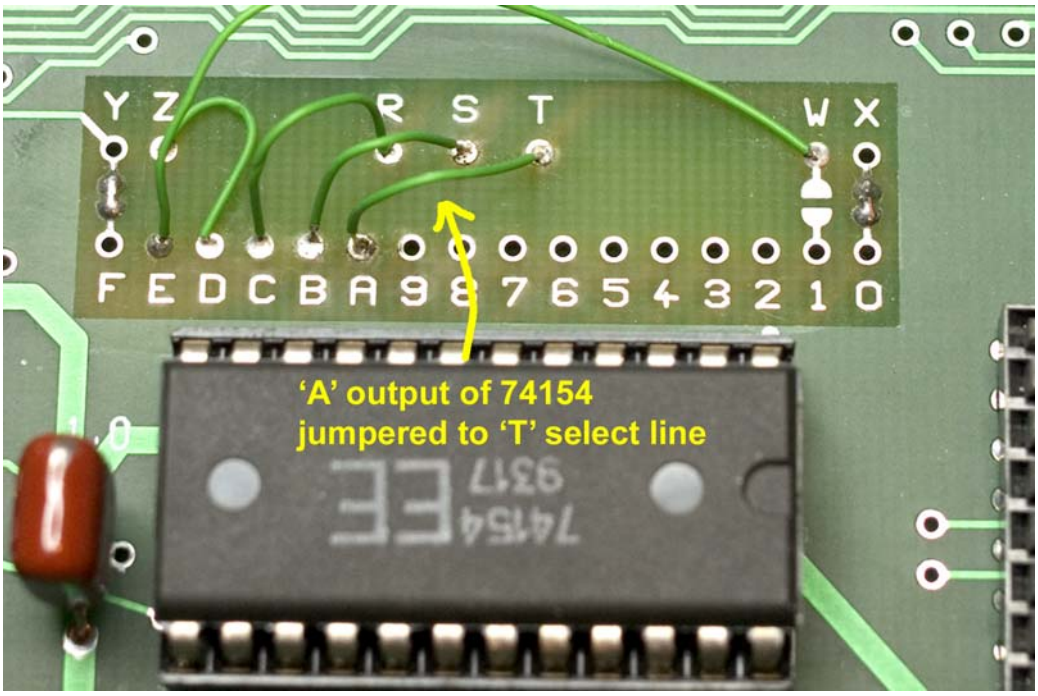


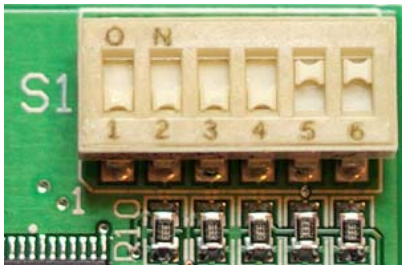
Figure 1: Typical mapping of Apple1 with the additional jumper for CFFA1.

2. Discharge yourself of excess static charge.
3. Remove the CFFA1 Interface from the anti-static bag.
4. Turn off power to your Apple1/Replica1 computer. **Do not skip this step!**
5. Configure dip switch S1 on the CFFA1 (upper right corner) for your computer. Table 1 gives the meaning of each switch. Figure 2 shows the typical configuration for the Apple1 and Replica1 computers. Note: Obtronix Apple1 clones should use the Apple1 settings.

Switch	Purpose	Apple1	Replica1
1	Not used	**X**	**X**
2	Disable SRAM decode from $1000 to $7FFF	**OFF**	**ON**
3	Low at CPLD input pin 19 (not used currently)	**X**	**X**
4	EEPROM Write Enable	**OFF**	**OFF**
5	SRAM Enable	**ON**	**ON**
6	Connects Apple1 Reset to CFFA1 CF card socket	**ON**	**ON**

Table 1: CFFA1 connections and jumpers PCB v1.0 Rev D
*NOTE: **ON** = switch up, **OFF** = switch down, and **X** = Don't Care*

Typical S1 settings for Apple1



Typical S1 settings for Replica1

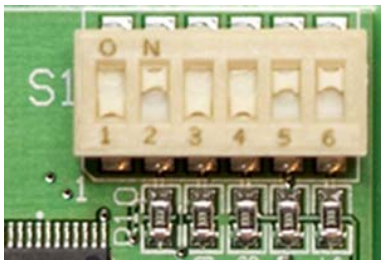


Figure 2: Typical dip switch configuration for the Apple1 and Replica1 computers.

6. Before Inserting the CFFA1 Interface into your computer, determine the proper orientation using the silk-screen labels: '**1**', '**22**', '**A**' and '**Z**' located near the gold edge card fingers. Match these labels up with the labels on the host computer before inserting the CFFA1. See Figure 3 below for a picture of the CFFA1 correctly inserted in the Apple1.

***** Warning: Inserting the CFFA1 card backward will destroy the CFFA1 *****

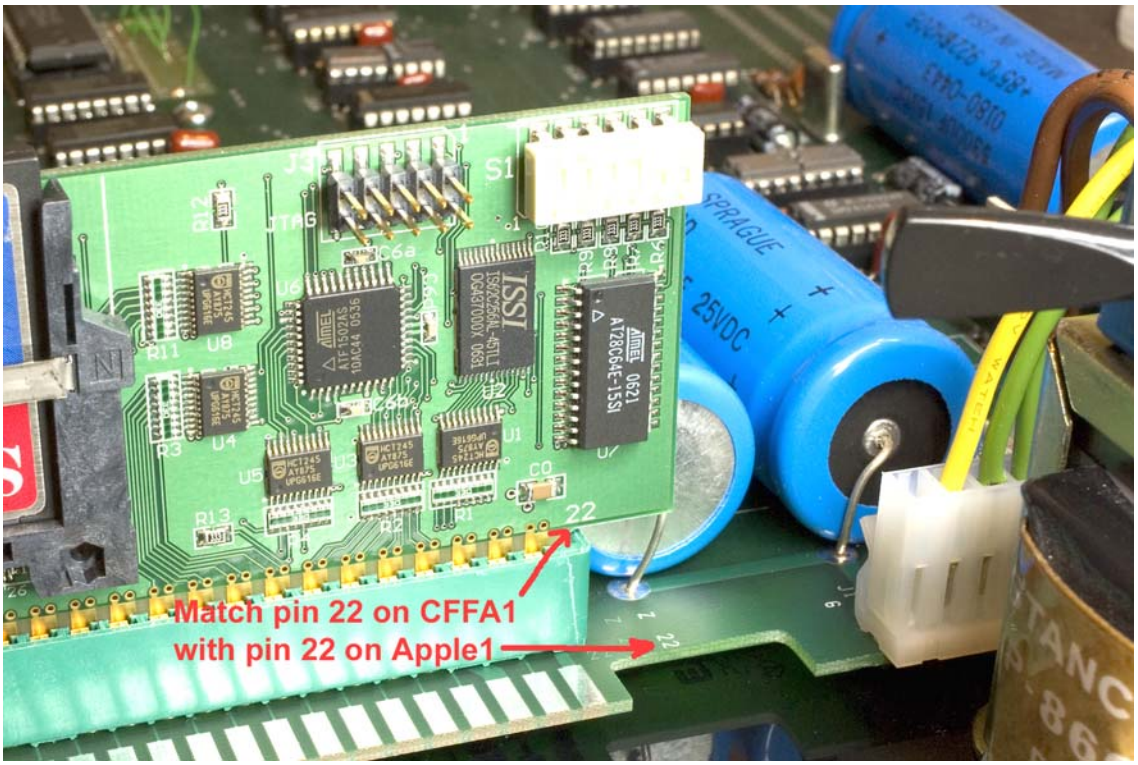


*Figure 3: CFFA1 inserted **correctly** in Obtronix clone of the Apple1.*

7. Insert the CFFA1 Interface into any empty slot. An authentic Apple1 or Obtronix clone only has one slot. A Replica1 computer will require an expansion board for the CFFA1 to work. All slots are wired in parallel so they should all work the same.
8. Insert a CompactFlash memory card into your CFFA1 interface. You can insert the card anytime, as long as you power cycle your computer after you insert the card. NOTE: Depending on the brand of CF card, you will either have to power cycle or press reset after each insertion of a CF card into the CFFA1.
9. Power up your Apple1 computer.

10. Press the clear screen button on the Apple1.
11. Press the Reset button on the Apple1.
12. Type 9000R to access the CFFA1's interactive menu. The firmware for the CFFA1 is located at $9000 to $AFFF.

Note to Replica1 Users:

You may need to connect a short ground wire (2 inches or less) from the Replica1 to its expansion board to get reliable operation from the CFFA1 (see Figure 4). The symptom you may notice is an unexpected I/O error when accessing the CF card from a simple Catalog menu command. If you get I/O errors, try adding the ground wire. The best connection will be a soldered connection, but other temporary connections may work also.

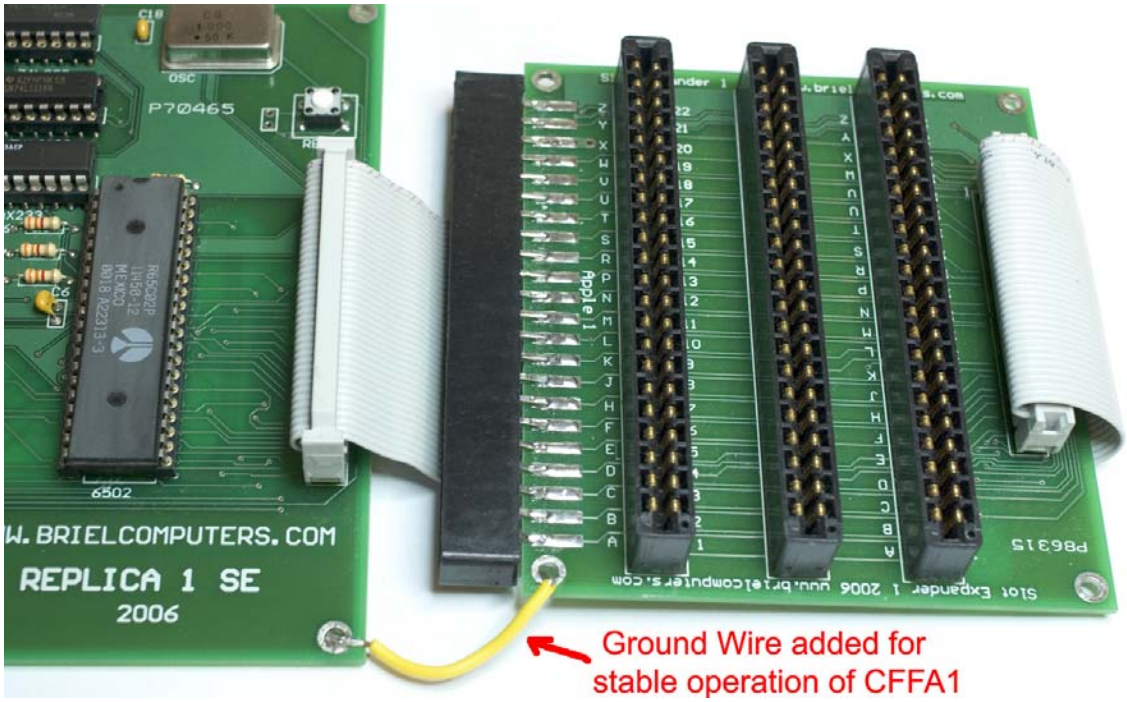


Figure 4: Replica1 SE and expansion board with ground wire added

Installation Details

The CFFA1 Interface card comes in an anti-static bag. This bag is a good place to store the card at times when the card is not installed in a computer. Before opening the zip-top bag, be sure you do not have a static charge built up in your body. The CFFA1 Interface card can be installed in the Apple1's expansion slot or in a Replica1 expansion interface board. To determine the correct cabling for the Replica1 and its expansion board, look on the bottom of the boards to find the square pin that denotes pin 1 of the connectors.

The following figure and tables define the location and meaning of all of the connectors and switches on the CFFA1 card. Use these tables to set the switches for CFFA1 operation in your Apple1 computer.

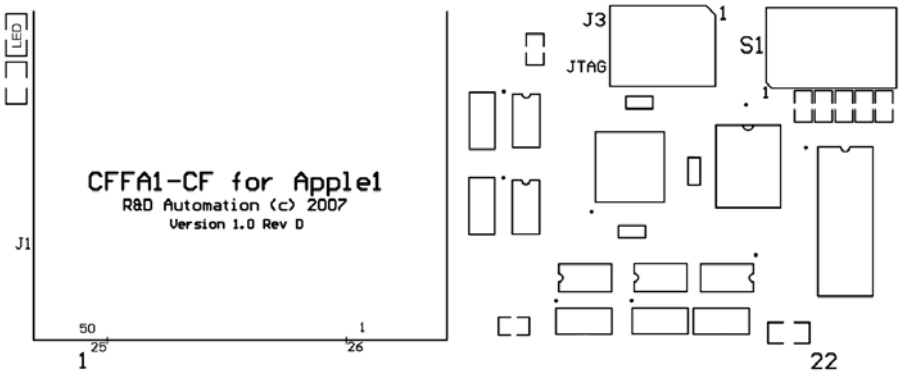


Figure 5: CFFA1 switches and connectors on PCB v1.0 RevD

Name	Purpose
J1	CompactFlash Socket for CF card or Microdrive.
J2 (not shown)	CFFA1 edge card connector. Plugs into an Apple 1 expansion slot. Note: the pin number designations **1** on the left and **22** on the right. On the back of the board you will find **A** and **Z.** Be sure these pin labels correspond when inserting your CFFA card into your Apple1 expansion slot.
J3	JTAG programming header for CPLD U6. Note: Power must be supplied to board during programming with this port. In other words, plug CFFA1 into an Apple 1 while using this port.

Table 2: CFFA1 headers, sockets, and edge connectors

Name	Purpose
S1-1	Unused. Can be in any position
S1-2	Set switch to ON to disable response from the onboard SRAM in the address range of $1000 to $7FFF. This is useful if you are using a Replica1 computer, which already provides SRAM in this address range. Leaving this switch OFF on a Replica1 or any host system with RAM in this address range will result in a bus fight and unpredictable computer operation.
S1-3	Set switch to ON to set pin19 of the CPLD to a logic 0. This pin is currently unused. But may be used in future release of CPLD logic firmware.
S1-4	EEPROM Write Enable. When set to OFF, this switch prevents accidental and intentional writes to the EEPROM. Always leave this switch in the OFF position until you want to change the EEPROM contents.
S1-5	Set switch to ON to enable the 32KB SRAM. This SRAM is mapped into the host computer at $1000 to $8FFF. This cannot be changed without reprogramming the CPLD. You can however prevent the SRAM from responding to addresses $1000 to $7FFF using S1-2. This is useful if you are using a Replica1 computer. Note: the Menu firmware developed for the CFFA1 needs RAM to exist from addresses $8000 to $8FFF on the host computer. If your host computer already has this RAM then disable the CFFA1's onboard SRAM by switching this switch OFF.
S1-6	Reset Enable. Factory Default: jumper installed. Connects Apple bus reset signal to CF device reset. I recommend leaving this jumper in.

Table 3: CFFA1 dip switch meaning

CFFA1 Partition Scheme

The CFFA1 Interface card uses the same fixed partition scheme supported by the CFFA for Apple *II*. The initial 1.0 release for the CFFA1 menu firmware only supports access to Drive 0 the first 32MB* partition on your CF card. Future updates to the CFFA1 onboard firmware may support access to more partitions on the CF card.

** Please note that the marketing departments of most device manufactures now define "MB" to mean 1,000,000 bytes instead of the more proper 1024 x 1024 = 1,048,576 bytes which the CFFA1 uses.*

Preparing the Storage Device

The menu firmware programmed on the CFFA1's EEPROM has support to read and write a CF card with the ProDOS file system. This means that you can take CF cards from your Apple *II* CFFA and read and write them in your Apple1. Please keep in mind that the CFFA1's firmware is not running ProDOS or any code from ProDOS. It was developed by Dave Lyons to parse the ProDOS file structures. The firmware can load and save Apple1 binary files and Woz Basic files. It can also format the first 32MB partition on your CF card. If you have access to an Apple *II* and a CFFA card (for Apple*II*), you may want to format your CF card in your Apple *II* using a ProDOS formatting tool, like Copy II+ or Davex. This will make your CF card bootable in your Apple II and accessible in your Apple1. If you format your CF card in your Apple1, the CF will not be bootable in your Apple *II*, because the CFFA1 format command does not write the ProDOS bootloader on to the CF card.

CF Removability

Although most CF cards are used in a "removable" sense, the CFFA1 interface card does not treat a CF card as a removable device. In other words, if you want to remove the CF card from the CFFA1 interface card, shut down your computer first.

Removing the card with the computer's power on will not hurt the CF card, but if you plug the card back in, you will not be able to access the data until you do a complete power cycle of the computer or a reset* of the card. The reason the CF card is not "removable" is that it is being used in the "True IDE" mode and should be thought of as a normal hard drive. For the same reason you don't pull out your hard drive with the power on, you should not pull out a CF card with the power on.

**Reset of the card will occur when the Apple performs a reset, as long as the CFFA1's reset enable switch S1-6 is up (on). A reset only works to reinitialize SOME brands of CF card back into TrueIDE mode. Some brands will only work after a power off/on cycle. SanDisk brand cards do support entering the TrueIDE mode when a reset occurs.*

Devices Compatible with CFFA1

The CFFA1 Interface card was developed using SanDisk CompactFlash cards. A wide variety of brands work with the card, but I can't guarantee compatibility. To help determine which devices work with the CFFA1 card and which devices do not, I will maintain a compatibility list on my web site, <http://dreher.net/CFforApple1>

If you have information about the compatibility of a storage device with the CFFA1 Interface card, feel free to post a message to the Discussion Forum or E-mail me about it. I will update the Compatibility list as information becomes available.

Auto-Booting from CF cards?

NO! There is no support in a real Apple1, or authentic clone, to support booting an OS. Remember this was year 1976 and the dawn of the Microcomputer revolution. The Apple1 had 256 bytes of PROM, held on two 4bit x 256 memories. This provided just enough space to hold a small monitor and not much else. Only through custom monitor modifications could you create an auto-booting Apple1.

Menu Firmware Details

This section describes in detail the features of the firmware available onboard the CFFA1. The firmware provides both the low-level block access to the cards and a high level menu to allow you to easily access the ProDOS file system. The CFFA1 firmware does not attempt to provide all ProDOS services and should not be thought of as an operating system. It is a menu driven program that can read and write the ProDOS file system.

For more information about the firmware's entry points, see Firmware section in the Advanced Information portion of this manual.

To access the interactive menu on the CFFA1 from Woz monitor simply type:

9000R

Alternately, if you are mainly using Woz BASIC and you want the Quit 'Q' command to always exit to Woz BASIC, then type:

9003R

In either case this will display a menu similar to the one shown below.

```
 CFFA1 MENU (1.0)
 ----------
 C - CATALOG      P - PREFIX
 L - LOAD         N - NEW DIRECTORY
 S - SAVE (BASIC) W - WRITE FILE
 R - RENAME       D - DELETE
^F - FORMAT       T - TERSE
 B - READ BLOCK   M - MEMORY DISPLAY
 Q – QUIT
```

Note: While being prompted for input from the various menu commands, the backspace key will work. You do not need to use the "_" (underscore) character like you do in the resident (Woz) monitor. To access the format command (^F) type <CTRL> F, where the '^' symbol indicates <CTRL>.

The current active directory can be changed using the "P" – Prefix command. You may notice a new file type BA1 when viewing a catalog. This has been defined as an Apple1 Basic or "WOZ" Basic file. It is not compatible with Applesoft or Integer basic files on the Apple II. The hex value for this type is: $F1.

CFFA1 Menu Firmware Commands

Command	Key	API Index	Description
Catalog	**C**	**N/A**	The Catalog command will display the contents of the current directory in a column format reminiscent of the ProDOS "CAT" command.
Details			
After each page of files, the catalog command will pause and wait for the user to enter <SPACE> or <CR> to continue the catalog listing. Alternately you can press <ESC> after each page to abort the catalog command.			

Command	Key	API Index	Description
Load	**L**	**$22, $26**	The Load command is used for loading any type of file into the Apple1 memory.

Details

The firmware handles Woz BASIC (BA1) files specially. All other file types are loaded into RAM in a single block. If the file you are BA1 type file, the load command will automatically load both the main program and the zero page block at: $4A to $FF. Load will prompt you for the file name,

`LOAD FILE:` (enter a standard ProDOS compatible file name)

Next you will be prompted for the starting address to load the file at. If the file has a load address already defined, it will be displayed in the $xxxx field shown below. If that "default" address is satisfactory, press <ENTER> to accept it. Note: if you are loading a BA1 basic file, you will not be prompted for the ADDR, as it is already known.

`ADDR ($xxxx):`

If the file loads is successful, the message `00 SUCCESS` will be displayed.

Important: Version 1.0 CFFA firmware loads data in multiples of blocks (512 bytes). So, even for a 1 byte file, 512 bytes are loaded. This limitation may be removed in future version of firmware.

Command	Key	API Index	Description
Save (BASIC)	**S**	**$24**	The save command is used specifically to save WOZ basic programs you have created or modified using the WOZ basic typically located at: $E000.

Details

When this command executes, it will save the two ranges of memory. The first range is the zero page locations used by Basic ($4A to $FF) and the BASIC program defined by HIMEM (4A-4B) and LOMEM (4C-4D). You do not typically need to set HIMEM or LOMEM manually to use this command.

You will be prompted to enter a name for the WOZ Basic program you are trying to save:

`SAVE:`

Enter the name you would like to use, if the save operation is successful the message `00 SUCCESS` will be displayed.

IMPORTANT: The Save command will not warn you if you are about to over-write an existing file on the CF card.

Command	Key	API Index	Description
Write	**W**	**$20**	The Write File command is used for saving all file types to the CF card except Basic files.

Details

The write command will prompt you for the starting address and the length of the data you want to write to the CF card. Then it will prompt you for a file name. This file must be a ProDOS compatible name of 15 characters or less.

Enter the starting address to start writing from:

```
WRITE FROM: $
```

Enter the length in bytes in hexadecimal

```
    LENGTH: $
```

Enter the file type. This is a 2 digit hex value of ProDOS file types.

```
TYPE (BIN): $
```
(press <ENTER> to accept BIN type)

Enter the name you want the file stored under in the CF card. Standard ProDOS naming rules apply.

```
      NAME:
```

If the write operation is successful the message `00 SUCCESS` will be displayed.

IMPORTANT: The Write command will not warn you if you are about to over-write an existing file on the CF card.

Command	Key	API Index	Description
Prefix	**P**	**N/A**	Prefix (Change Directory) The Prefix command is the ProDOS way of saying change directory.

Details

To change to the top level (root) directory simply press <enter> when prompted for the directory name. To select a subdirectory enter the name of that directory.

```
PREFIX DIR:
```

Enter the name of the new directory you want to change to.

IMPORTANT: CFFA1 firmware v1.0 allows a single level of directories. You can create directories within the root directory, but you can't have directories inside those directories. However, each directory can hold as many files as available disk space allows. The root directory can only hold 51 files.

Command	Key	API Index	Description
New Directory	**N**	**$26**	The New Directory command allows you to create a new directory in the file system.

Details

The command will prompt you for the name of the new directory.

```
NEW DIRECTORY:
```

Enter the name of the directory you want to create.

IMPORTANT: Currently there is a nesting limit of one directory level so all new directories must be created in the root (top level) directory.

Command	Key	API Index	Description
Rename	**R**	**$28**	The Rename command allows you to change the name of a single file or directory.

Details

The command will prompt you for the name of the file to rename. The new name must not already exist in the current directory.

```
RENAME:
```

Enter the name of the file you want to rename.

```
   TO:
```

Enter the new name you would like the file to be called.

Command	Key	API Index	Description
Delete	**D**	**$2A**	The Delete command allows you to delete a file or empty directory from the ProDOS file system.

Details

The command will prompt you for the name of the file or empty directory to be removed. Although the file is removed from the file system, the data blocks that stored the file's content are not overwritten.

```
DELETE:
```

Enter the name of the file you wish to delete.

Command	Key	API Index	Description
Format	**<CTRL> F**	**$2E**	The Format command writes the ProDOS file system to the CF card.

Details

The Format command allows you to initialize the CF card for operation in the CFFA1. Format performs the following actions:

1) Write Zeros to every byte in CF blocks 0 to 64. This will complete erase any existing file system and master boot record on the CF card.
2) Constructs and writes block 2, the main ProDOS directory block.
3) Constructs and writes blocks 3, 4, and 5, which just contains links to other blocks.
4) Constructs and writes the volume bit map starting at block 6. Adding 1 block for every 2MB (or fraction thereof) of volume size. For a 32MB volume, the bitmap takes 16 blocks, from 6 to 21.

```
DESTROY DATA ON DRIVE 0
DESTROY VOLUME: <Volume Name>
   BLOCKS USED: <Volume Size>
ARE YOU SURE? ('YES')
```

Enter “YES” here if you want to format the CF card’s first partition. Entering anything else will abort the format command.

IMPORTANT: If you are planning to use your CF card in your Apple II and your Apple1, I recommend that you format your CF cards in your Apple II. By formatting the card in your Apple II you will be able to boot your CF in your Apple II if you chose. If you only use your CF card in your Apple1, then the menu format command will work fine.

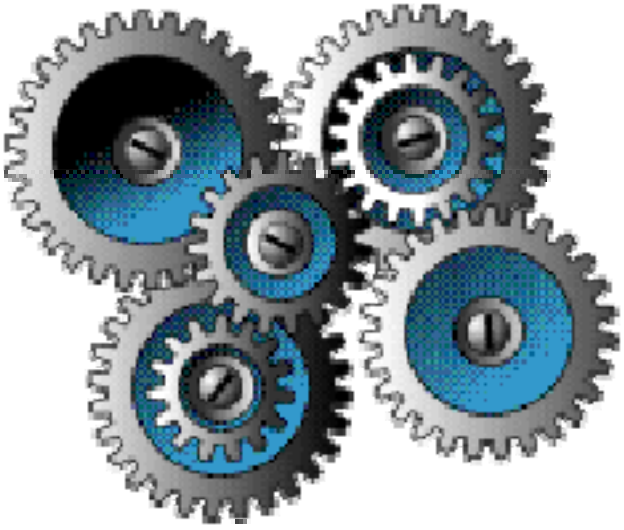
Command	Key	API Index	Description
Terse	**T**	**N/A**	The Terse command simply toggles the menu display on and off.
Details			
This is useful due to the slow rate at which characters display on the Apple1 and can save you a lot of time. While in terse mode, if you press a key that is not recognized by the menu firmware, the menu will be displayed once, but you are still in terse mode. When you press "T" the message `TERSE ON` OR `TERSE OFF` will be displayed to show the new mode.			

Command	Key	API Index	Description
Read Block	**B**	**N/A**	The Read Block command allows you to read any 512-byte block on the CF card without regard to the type of data it is.
Details			
The data block will be displayed in hexadecimal and ASCII format. Once the command is complete the data will be left at address range: $8A00 - $8BFF, until that space is used by another command. *Note: this address is subject to change in future releases of firmware.*			

Command	Key	API Index	Description
Display Memory	**M**	**N/A**	The Display Memory command allows you to view the content of any memory in the Apple1.
Details			
The data is displayed in hexadecimal and ASCII format.			

Command	Key	API Index	Description
Quit	**Q**	**N/A**	The Quit command will cause the menu firmware to stop executing.
Details			
Which memory address the Quit command returns to is dependant on which entry point you used to start the menu command. For more information about the firmware's entry points, see *Table 5: CFFA1 Firmware Entry points.*			

Advanced Information



Hardware

This section gives detailed information about the hardware used on CFFA1 Interface card.

CPLD

The chip U6 is an Atmel ATF1502AS-10AC44 part, which is compatible with an Altera EPM7032STC44-10 CPLD (Complex Programmable Logic Device). It is flash based and can be reprogrammed via the JTAG Port J3. For this you will need Altera compatible programming cable and Atmel's ATMISP program for Windows. When programming the CFFA1 card's CPLD via the JTAG header, the CFFA1 card must be plugged into an Apple 1 bus to supply power to the card. I do not recommend programming the CPLD with a CF card installed.

Programming cables are available from many places, including eBay, for as little as US$15. Search for "ByteBlaster Cable". The older 5 volt cables will work fine. The more expensive low voltage cable is not needed but may also work.

Atmel CPLD Pinout

Figure 6 provides the signal names for version 1.0 of the CFFA1 CPLD firmware (here "firmware" refers to the AHDL logic files used to program the CPLD).

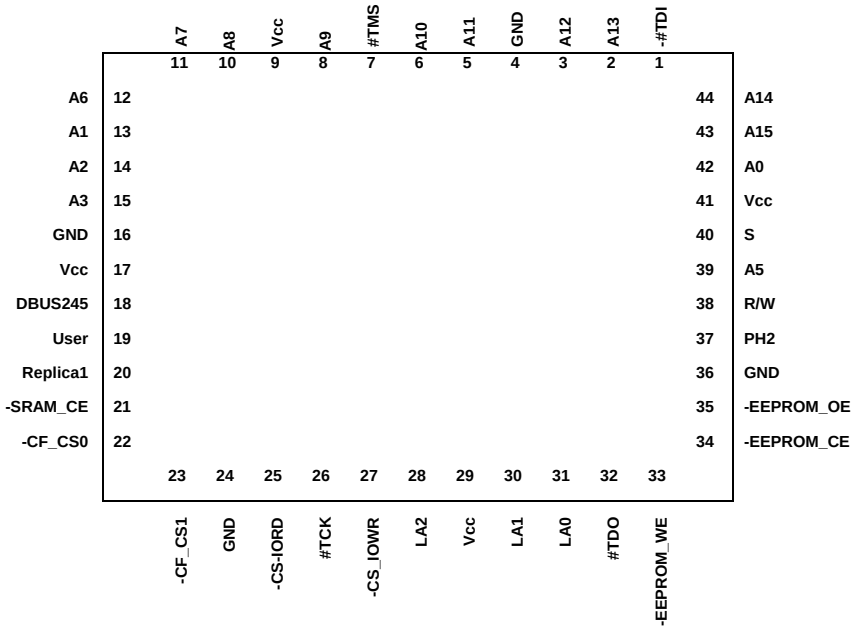


Figure 6: Atmel's ATF1502AS-10AC44
NOTE: Signal names are specific to CFFA1 logic

CPLD Memory Mapping

The logic resident on the CPLD creates the memory map shown in Table 4. This mapping was programmed into CPLD after the CFFA1 was assembled. The exact logic file used to create this mapping is shown in the next section. Changing this mapping will require reprogramming the CPLD via J3 and a JTAG programmer like an Altera Byte Blaster.

NOTE: Changes to this layout may require changes to the CFFA1's EEPROM menu firmware also.

Address	CFFA1 Memory Mapping	
$1000 $7FFF	U2 - 32KB SRAM	**Switch S1-2:** enable/disable the region $1000 - $7FFF region **Switch S1-5:** enable/disable the entire SRAM chip.
$8000 $8FFF		
$9000 $9FFF	U7 – EEPROM	Always mapped in, when CFFA1 is inserted
$A000 $AFFF		Enabled during accesses to $A000 range as long as the slot signal “T” (memory select line on pin “L”) is low. Mapping T to any other address will cause this 4KB portion of firmware to disappear.

Table 4: CPLD Memory Map

CPLD Logic

Listing 1 is the ADHL source file used to create the programmer-ready .jed file needed to program U6. Comments in the file start and end with the % character. This file was compiled using Altera’s Quartus 6.0 sp1 Web edition development software and then converted to a .jed file using the Pof2Jed utility. This software is free and can be downloaded from Altera’s and Atmel web sites. A free license from Altera is needed to use the Quartus software.

Listing 1: CPLD Logic - AHDL Source

```
%--------------------------------------------------------------------------------
 CompactFlash Interface for the Apple 1 computer
 Project Home: http://dreher.net/CFforApple1/
 Project Version 1.0   April, 2007

 Version 0.8  -  Prototype 3: CF card in TrueIDE mode
 Version 0.9  -  Production RevD: EEPROM was being selected at same time
                 as CF card for CF addresses
 Version 1.0  -  Initial Release
--------------------------------------------------------------------------------%

SUBDESIGN CFLogic
(
  A0, A1, A2, A3, A5, A6, A7, A8, A9    :INPUT;
  A10, A11, A12, A13, A14, A15          :INPUT;
  PH2, /EN_T, R/W, User, /Replica1      :INPUT;

  /SRAM_CE                              :OUTPUT;
  /CF_CS0, /CF_CS1, /CF_IORD, /CF_IOWR  :OUTPUT;
  /EEPROM_WE, /EEPROM_CE, /EEPROM_OE    :OUTPUT;
LA0, LA1, LA2, /DBUS245                 :OUTPUT;
)

VARIABLE
/EEAddr, /EEAuxAddr                     :NODE;
/SRAMMainAddr, /SRAMAuxAddr             :NODE;
/CFAddresses                            :NODE;
/RESET_MASK, /SET_MASK, CS_MASK         :NODE;
```

```
LA3                                   :NODE;

BEGIN

%------------------------ 8K EEPROM Control logic -------------------------------%

% EEPROM Address: 4K direct mapped to $9000 to $9FFF %
 /EEAddr = !A15 # A14 # A13 # !A12 # !/CFAddresses;

% EEPROM Address: 4K mapped to T select line at $A000 to $AFDF as long as Select
  line T is connected to A on Apple1%
% Note: The CF card is mapped into $AFF0-$AFFF and $AFE0-$AFEF (It is duplicated
  twice because A4 is not available for address decode) %
 /EEAuxAddr = /EN_T # (!A15 # A14 # !A13 # A12) # !/CFAddresses;

% EEPROM Chip Enable %
 /EEPROM_CE = (/EEAddr !$ /EEAuxAddr) # !PH2;


% EEPROM Output Enable %
 /EEPROM_OE = !R/W # /EEPROM_CE;

% EEPROM Write Enable %
 /EEPROM_WE  = R/W # /EEPROM_CE;

%------------------------ 32K SRAM Control logic -------------------------------%

% 1) Decode SRAM address at: $1000 to $7FFF. Assumes there is 4K DRAM already on
     board Apple 1 at $0000%
% /Replica1 =0 to disable SRAM on CFFA1.%
 /SRAMMainAddr = A15 # (!A14 & !A13 & !A12);

% 2) Decode SRAM address at $8000 to $8FFF %
 /SRAMAuxAddr = !A15 # A14 # A13 # A12;

% If Replica1 then disable SRAM from $1000 to $7FFF. %
% If Apple1   then  enable SRAM from $1000 to $7FFF. %
% In both Apple1 and Replica1 always map 4K sram at $8000 to $8FFF %
 /SRAM_CE = ((/SRAMMainAddr  # !/Replica1) !$ /SRAMAuxAddr) # !PH2;

% Pass Address line through, for now, unlatched %
  LA0 = A0;
  LA1 = A1;
  LA2 = A2;
  LA3 = A3;

%-------------------------------------------------------------------------------%
% Fix for SanDisk Family of CompactFlash drives. True IDEmode is not quite     %
% True! The idea here is to mask the read cycle that proceeds all write cycles %
% because the read cycle was confusing the Sandisk                              %
%-------------------------------------------------------------------------------%
% SetMask = $AFF1   %
 /SET_MASK = /CFAddresses # (LA3 # LA2 # LA1 # !LA0);
% ResetMask = $AFF2 %
 /RESET_MASK = /CFAddresses # (LA3 # LA2 # !LA1 # LA0);
  CS_MASK  = SRFF(!/SET_MASK, !/RESET_MASK, PH2, VCC, VCC);

%----------------------------- CF Control logic --------------------------------%
% decode CF card address range at $nFF0 to $nFFF. Because Address lines A4 is not
  available to this logic, the CF address space will repeat from nFE0-nFEF, where
  n is the 4K space mapped to select line S  %
 /CFAddresses = /EN_T  # !A11 # !A10 # !A9 # !A8 # !A7 # !A6 # !A5;

% CF Chip Select0 line: addresses nFF8 to nFFF %
 /CF_CS0 = /CFAddresses # !LA3 # (CS_MASK & R/W);
% CF Chip Select1 line: addresses nFF6 to nFF7 %
 /CF_CS1 = /CFAddresses # (LA3 # !(LA1 & LA2)) # (CS_MASK & R/W);

 /CF_IORD  = !R/W # /CFAddresses # !PH2;
 /CF_IOWR  =  R/W # /CFAddresses # !PH2;

 /DBUS245 = /EEPROM_CE & /SRAM_CE & /CF_CS0 & /CF_CS1;

END;
```

Firmware

This section gives detailed information about the EEPROM based code (firmware) shipped with the CFFA1 Interface card.

User Accessible Firmware Entry Points

The firmware resident on the CFFA1 EEPROM provides a variety of firmware entry points for the user, as shown in Table 5. The first two entry addresses, $9000 and $9003 will be the most useful to a typical CFFA1 user. The key difference between these entry points is in what happens when you select the Q – Quit option on the menu.

Address	Purpose
$9000	Standard Menu Firmware entry point. Menu option 'Q' (Quit) exits back to Woz monitor at $FF1F.
$9003	Alternate Menu Firmware entry point. Menu option 'Q' (Quit) exits back to Woz BASIC (if loaded) at address $E2B3. If basic is not resident the Apple1 will likely crash. Note: The standard Woz BASIC entry point is $E000, which clears out any existing BASIC program. The alternate BASIC entry point $E3B3 is useful if you want to return to Woz BASIC and not erase any existing BASIC program. If writing a Woz BASIC program from scratch (not loading one from the CF card) you should enter BASIC using E000R entry point. Then use CALL –24000 to reach the CFFA menu.
$A240	Jumps to the $9003 entry point. This entry point can be used from Woz BASIC by typing CALL –24000. This is easy to remember and when you quit the menu, you will return to Woz BASIC via the $E3B3 BASIC entry point.
$9006	Alternate Menu Firmware entry point. Q (Quit) will RTS back to caller. This entry point is designed to be used via JSR $9006.
$9009	Entry point for low-level CF block driver code. This code must be called with a JSR $9009, and specific Zero-page locations must be set up before being called. See firmware source for additional information
$900C	Entry point for access to CFFA1 firmware Application Programming Interface (API). A simple set of functions used by the CFFA firmware is exposed via this entry point. Prior to calling this entry point, various zero page locations must be initialized and the X register must be loaded with the requested API function to be executed.

Table 5: CFFA1 Firmware Entry points.

Firmware Updates

All new firmware updates will be made available at the CFFA1 web site: <http://dreher.net/CFforApple1/> in the "downloads" section. They will be available as both ASCII and binary files. If you have access to Andy McFadden's CiderPress program for Windows and your CFFA1 is working well enough to load in the new image from your CF card, then you can use a small FLASH programming utility that makes updating your CFFA firmware very easy. If your CFFA1 firmware has become corrupted, you may need to use a serial terminal, if available, for your Apple1 to load the flash image.

Below are detailed instructions describing how to load version 1.0 CFFA1 firmware into the CFFA1's EEPROM using a FLASH.BIN utility. These instructions assume you have a working copy of Andy McFadden's CiderPress for Windows on a PC with a CF card reader recognized by Windows.

Loading CFFA1 firmware using CiderPress

On your PC:

- Insert CF card into card reader on your Windows PC.
- Run CiderPress
- Click File -> Open Volume
- Uncheck the "Open as Read-Only" check box to allow writing.
- Select the physical disk that corresponds to the CF card and click OK

- Now go to “Actions” -> “Add Files” (if it is grayed out, you forgot to uncheck the "Open as Read-Only" check box above.

- If “Select location” dialog appears, select the first volume on the CF card. (The single volume limitation may be removed in the future.)

- From the “Add Files…” dialog, find and select the files: CFFA1V1.0.BIN and FLASH.BIN for your PC hard drive, or where ever they are located. The default settings for the other options on this dialog will work fine. Click Accept.

- Now that the two files are on the CF card, edit their attributes one at a time, by right clicking on the files and selecting “edit attributes".

- On the “Edit Attributes” dialog, change the "Aux Type" (hex):" entry from $0000 to the following. The File type should already be $06, if not, make the necessary changes.

$2000 for CFFA1V1.0.BIN

$1000 for FLASH.BIN

This updates the Aux type information in the files directory entry. Now the firmware will know where to load these files.

- Close the volume by selecting “File -> Close from the menu.

- Remove the CF card from your PC.

On your Apple1:

- Insert CF card in Apple1.

- Power up Apple1 computer.

- Run exiting CFFA1 menu firmware:

```
9000R
```

- In the menu select the Load menu option to load in CFFA1V1.0.BIN

- Accept the default address if it is $2000. If not entry 2000 <enter>.

- Select the Load menu option again and load in FLASH.BIN

- Accept the default address if it is $1000. If not entry 1000 <enter>.

- Quite the CFFA1 firmware back to Apple1 monitor.

- Type: **`1000R`** (to run the FLASH program)

- Follow instructions on screen. (Basically just flip switch S1-4 up, and hit a key to begin programming. Which takes about 4 seconds.)

If all has gone well, the FLASH program should program and verify the new firmware has been copied from RAM at $2000 to EEPROM at $9000. If there is a verify failure, check the dip switch S1-4 to be sure it is set to enable writes to the EEPROM.

- Flip switch S1-4 down to write protect the EEPROM.

Contributing Firmware to the CFFA1 Project

There are two ways in which you can contribute to the firmware portion of this project.

1. You can send me your ideas for improvements that I will consider integrating into a future firmware version.
2. Write your own firmware/driver for my hardware and send me the working source code and binary, and I will post it on my web site under a contributors’ section.

EEPROM Layout

The chip U7 located at the right of the board is an Atmel AT28C64E-15SI 64Kbit EEPROM. The E in the part number stands for high write endurance of 100K re-writes.

The CPLD logic decodes the EEPROM at address $9000 to $AFFF. Although the last 32 bytes of this range are reserved for the CF card, making those EEPROM locations inaccessible by the EEPROM. Even though this address mapping is statically defined in the CPLD, the CFFA1 also requires that the Apple1's slots 'T' select line be connected to the "A" output of the Apple1 74154 decoder chip.

To make decoding easier, the two halves of the 8K address space were mapped into the Apple1 address space in reverse order. This has no effect when using or re-programming the EEPROM while it is in circuit. The reverse mapping is completely transparent to the user. However, if you were going to program the EEPROM in an external programmer like I was doing early in the project development, you would need to be aware that physical EEPROM addresses $0000 to $0FFF are mapped to $A000 and $1000 to $1FFF are mapped to $9000.

CFFA1 CompactFlash Memory Mapped I/O

Table 6 shows all of the CF I/O addresses decoded by the CFFA1 Interface card. These addresses are used to interface a CF card's task register file to the Apple's bus. There are also two special soft switches used to inhibit CPU read cycles from confusing CF cards during block write routines. These were added because some CF cards are not tolerant of read-cycles during PIO block writes in "TrueIDE" mode.

NOTE: Because the CPLD on the CFFA1 doesn't have address line A4 connected to it, the memory map in table 6 below repeats at addresses $AFE0 to $AFEF also. To maintain compatibility with any future version of the CFFA1 don't use this mirrored range of addresses.

Apple 1 Address	Name Used in Source Code	Read/ Write	Description
$AFF0	ATAData	R/W	This register is used to transfer data between the host and the attached device.
$AFF1	SetCSMask	R/W	Special soft switch to **disable** CS0 & CS1 signaling to attached device during 65C02 read cycles that always precede write cycles
$AFF2	ClearCSMask	R/W	Special soft switch to **enable** CS0 & CS1 signaling to attached device during 65C02 read cycles that always precede write cycles
$AFF3			Unused
$AFF4			Unused
$AFF5			Unused
$AFF6	ATADevCtrl	W	This register is used to control the device's interrupt request line and to issue an ATA soft reset to the device.
$AFF6	ATAAltStatus	R	This register returns the device status when read by the host. Reading the ATAAltStatus register does NOT clear a pending interrupt. (NOTE: CFFA1 does not use interrupts)
$AFF7			Unused
$AFF8			Unused
$AFF9	ATAError or ATAFeature	R	This register contains additional information about the source of an error when an error is indicated in bit 0 of the ATAStatus register.
$AFFA	ATASectorCnt	R/W	This register contains the number of blocks of data requested to be transferred on a read or write operation between the host and the device. If the value in this register is zero, a count of 256 sectors is specified.
$AFFB	ATA_LBA07_00	R/W	This register contains the starting sector number or bits 7-0 of the Logical Block Address (LBA) for any device access for the subsequent command.
$AFFC	ATA_LBA15_08	R/W	This register contains the low order 8 bits of the starting cylinder address or bits 15-8 of the Logical Block Address.
$AFFD	ATA_LBA23_16	R/W	This register contains the high order bits of the starting cylinder address or bits 23-16 of the Logical Block Address.
$AFFE	ATA_LBA27_24	R/W	This register is used to select LBA addressing mode or cylinder/head/sector addressing mode. Also bits 27-24 of the Logical Block Address.
$AFFF	ATACommand	W	A write to this register will issue an ATA command to the device.
$AFFF	ATAStatus	R	This register returns the device status when read by the host. Reading the Status register does clear a pending interrupt. (NOTE: CFFA1 does not use interrupts)

Table 6: CF card address space defined by the CFFA1 Interface card

Application Programming Interface (API)

A file called CFFA1_API.s is available to programmers who would like to gain access to the CFFA1 firmware from their own programs. The file contains equates a programmer will need, including the entry point addresses for the various CFFA1 firmware routines.

Listing 2: Programmers API Info

```
;-------------------------------------------------------------------------------
; CFFA1_API.s  Version 1.0 - 05/22/2007
;
; Equates for calling the CFFA1 API -- Firmware version 1.0 ($01)
;-------------------------------------------------------------------------------

CFFA1_ID1           = $AFFC   ; contains $CF when CFFA1 card is present
CFFA1_ID2           = $AFFD   ; contains $FA when CFFA1 card is present

FirmwareVersion     = $01

;-------------------------------------------------------------------------------
; Entry points to the CFFA1 firmware:
;
; MenuExitToMonitor
;   JMP here to display the CFFA1 menu.
;   Quit puts the user into the monitor.
;
; MenuExitToBASIC
;   JMP here to display the CFFA1 menu.
;   Quit puts the user into BASIC.
;
; Menu
;   JSR here to display the CFFA1 menu.
;   Quit returns control to your code.
;
; CFBlockDriver
;   JSR here to read or write a block, after setting up pdCommandCode
;   and other inputs (see below).
;   Result:  CLC, A = 0
;            SEC, A = error code
;
; CFFA1_API
;   JSR here to call one of many functions provided by the firmware.
;   See "Function selectors for CFFA1_API" below.
;
;-------------------------------------------------------------------------------
MenuExitToMonitor   = $9000
MenuExitToBASIC     = $9003
Menu                = $9006
CFBlockDriver       = $9009
CFFA1_API           = $900C


;-------------------------------------------------------------------------------
; Inputs for CFBlockDriver - ProDOS block interface locations
;-------------------------------------------------------------------------------
pdCommandCode       = $42     ; see below
pdUnitNumber        = $43     ; always set this to 0 for firmware 1.0
pdIOBufferLow       = $44
pdIOBufferHigh      = $45
pdBlockNumberLow    = $46
pdBlockNumberHigh   = $47

;
; Values for pdCommandCode
;
PRODOS_STATUS       = $00
PRODOS_READ         = $01
PRODOS_WRITE        = $02
```

```
PRODOS_FORMAT       = $03


;-------------------------------------------------------------------------------
; Function selectors for CFFA1_API.
;
; Load one of these values into X:
;
;    ldx #CFFA1_xxxxx
;    jsr CFFA1_API
;
; Result:  CLC, A = 0
;          SEC, A = error code
;
; Certain functions have additional outputs, as described below.
;
;-------------------------------------------------------------------------------
;
; CFFA1_Version:
;   Output: X = current firmware version
;           Y = oldest compatible firmware version
;
; CFFA1_Menu:
;   Result: Runs the CFFA1 menu and returns when the user chooses Quit.
;
; CFFA1_DisplayError:
;   Input:  A = an error code
;   Result: Prints out a carriage return, the 2-digit hex error code,
;           and a description of that error, if available.
;
; CFFA1_OpenDir:
;   Input:  None (operates on the current prefix directory)
;   Result: Prepares for one or more calls to ReadDir.
;
; CFFA1_ReadDir:
;   Setup:  You have to call OpenDir before calling ReadDir.
;   Result: If no error, EntryPtr points to the next occupied directory entry.
;
; CFFA1_FindDirEntry:
;   Input:   Filename = name to search for
;   Result:  If no error, EntryPtr points at the found item's directory entry.
;
; CFFA1_WriteFile:
;   Input:  Filename = name for new file (will be replaced if it already exists)
;           Destination = starting address
;           FileSize = number of bytes to write
;           Filetype = type for new file
;           Auxtype = auxiliary type for new file
;
; CFFA1_ReadFile:
;   Input:  Filename = file to read into memory
;           Destination = starting address ($0000 to use the file's Auxtype value)
;
; CFFA1_SaveBASICFile:
;   Input:  Filename
;
; CFFA1_LoadBASICFile:
;   Input:  Filename
;
; CFFA1_Rename:
;   Input:  OldFilename = original name
;           Filename = new name
;
; CFFA1_Delete:
;   Input:  Filename = file or empty directory to delete
;
; CFFA1_NewDirectoryAtRoot:
;   Input:  Filename = name for new directory
;
; CFFA1_FormatDrive:
;   Input:  Filename = name for new volume
;           A = drive number (always set to 0 for firmware 1.0)
;           Y = $77 (just to help avoid accidental formatting)
;   Result: Disk volume is erased and given the specified name.
```

```
;
;------------------------------------------------------------------------------
CFFA1_Version            = $00
CFFA1_Menu               = $02
CFFA1_DisplayError       = $04

CFFA1_OpenDir            = $10
CFFA1_ReadDir            = $12
CFFA1_FindDirEntry       = $14

CFFA1_WriteFile          = $20
CFFA1_ReadFile           = $22
CFFA1_SaveBASICFile      = $24
CFFA1_LoadBASICFile      = $26
CFFA1_Rename             = $28
CFFA1_Delete             = $2A
CFFA1_NewDirectoryAtRoot = $2C
CFFA1_FormatDrive        = $2E


;------------------------------------------------------------------------------
; Zero-page inputs and results for API functions
;
; Filename and OldFilename point to strings that begin with a length byte (from
; 1 to 15), and each character must have its high bit off.  For example:
;
;    Filename   = $80     $280: 05  48  45  4C  4C  4F
;    Filename+1 = $02               'H'  'E' 'L' 'L' 'O'
;------------------------------------------------------------------------------
Destination         = $00                   ; 2 bytes
Filename            = Destination+2         ; 2 bytes
OldFilename         = Filename+2            ; 2 bytes
Filetype            = OldFilename+2         ; 1 byte
Auxtype             = Filetype+1            ; 2 bytes
FileSize            = Auxtype+2             ; 2 bytes
EntryPtr            = FileSize+2            ; 2 bytes


;------------------------------------------------------------------------------
;
; ProDOS low-level return codes
;
;------------------------------------------------------------------------------
PRODOS_NO_ERROR      = $00    ; No error
PRODOS_BADCMD        = $01    ; Bad Command (not implemented)
PRODOS_IO_ERROR      = $27    ; I/O error
PRODOS_NO_DEVICE     = $28    ; No Device Connected
PRODOS_WRITE_PROTECT = $2B    ; Write Protected
PRODOS_BADBLOCK      = $2D    ; Invalid block number requested
PRODOS_OFFLINE       = $2F    ; Device off-line
;
; High-level return codes
;
eBadPathSyntax       = $40
eDirNotFound         = $44
eFileNotFound        = $46
eDuplicateFile       = $47
eVolumeFull          = $48
eDirectoryFull       = $49
eFileFormat          = $4A
eBadStrgType         = $4B
eFileLocked          = $4E
eNotProDOS           = $52
eBadBufferAddr       = $56
eBakedBitmap         = $5A
eUnknownBASICFormat  = $FE
eUnimplemented       = $FF


;------------------------------------------------------------------------------
; ProDOS directory entry structure offsets
;------------------------------------------------------------------------------
oFiletype           = $10
oKeyBlock           = $11
```

```
oBlockCount         = $13
oFileSize           = $15
oCreateDateTime     = $18
oVersion            = $1C
oMinVersion         = $1D
oAccess             = $1E
oAuxtype            = $1F
oModDateTime        = $21
oHeaderPointer      = $25

oDirLinkPrevious    = $00
oDirLinkNext        = $02
oVolStorageType     = $04
oVolVersion         = $20
oVolAccess          = $22
oVolEntryLength     = $23
oVolEntriesPerBlock = $24
oVolFileCount       = $25
oVolBitmapNumber    = $27
oVolTotalBlocks     = $29

;
; ProDOS Storage types
;
kSeedling           = $10
kSapling            = $20
kTree               = $30
kExtended           = $50
kDirectory          = $D0
kSubdirHeader       = $E0
kVolume             = $F0
kStorageTypeMask    = $F0

;
; Filetypes
;
kFiletypeText       = $04
kFiletypeBinary     = $06
kFiletypeDirectory  = $0F
kFiletypeBASIC1     = $F1
kFiletypeBAS        = $FC
kFiletypeSYS        = $FF

;-------------------------------------------------------------------------------
; end of CFFA1_API.s
;-------------------------------------------------------------------------------
```

Contact Information

The following is a list of information resources for the CFFA1 Interface Card project. If you have questions, comments, or problems to report, please contact me using one of the methods listed below.

CFFA1 Web Site

The CFFA1 web site is located at: <http://dreher.net/CFforApple1/>. There you will find any new firmware revisions, project revisions, and general project status information.

Internet E-Mail

I can be reached via E-mail at: rich@dreher.net.

If you are reporting a problem, please use “CFFA1 problem“ or similar as your subject line. In your E-mail you should include the firmware version number, which can be determined by looking right after the words “CFFA MENU” when you run the menu firmware at $9000. Also, if possible, describe the conditions necessary to cause the problem.

CFFA1 Message Web Forum

To post a message in the forum, simply browse to: <http://dreher.net/phpBB/>. The CFFA1 message forum is a good place to post technical problems and solutions. Other users may have had similar problems and know of a possible solution. You will have to register before you can post to the forum the first time. I will have to manually approve your registration request. This is because of all the spam bots on the net. If you don’t get registered with a day, email me and ask if I approved you yet.

Acknowledgements

I would like to thank the following people for providing help on this project:

Sherry Dreher

Dave Lyons

Vince Briel

Philip Lord

Appendix 1: Firmware Listing

The following is a listing of the CFFA1 firmware located on the EEPROM. Please check for future firmware listings on the CFFA1 web site. The list file is included in each firmware archive. Thank you to Dave Lyons for developing the firmware for the CFFA1.

Listing 32: CFFA1 Firmware Version 1.0

```
ca65 V2.11.0 - (C) Copyright 1998-2005 Ullrich von Bassewitz
Main file   : CFFA1.s
Current file: CFFA1.s

000000r 1               ;--------------------------------------------------------------------------------
000000r 1               ; CFFA for Apple 1 firmware - CF Interface for Apple 1/Replica 1 computers
000000r 1               ; CFFA1.s  Version 1.0 - 05/22/2007
000000r 1               ;
000000r 1               ; Firmware Contributors:        Email:
000000r 1               ;   Dave Lyons                    dlyons@lyons42.com
000000r 1               ;   Rich Dreher                   rich@dreher.net
000000r 1               ;
000000r 1               ; This firmware is designed to execute on an orginal 6502
000000r 1               ;
000000r 1               ; Tools used to build this firmware: CA65: 6502 Cross Assembler
000000r 1               ;   http://www.cc65.org/
000000r 1               ;
000000r 1               ; Here is the copyright from that tool using --version option
000000r 1               ; ca65 V2.11.0 - (C) Copyright 1998-2005 Ullrich von Bassewitz
000000r 1               ;
000000r 1               ; Example build instructions on an MSDOS based machine:
000000r 1               ;-------------------------------------------------------
000000r 1               ; Assumes you have installed the CC65 package and set your path, etc.
000000r 1               ;
000000r 1               ; 1) ca65 --cpu 6502 -l CFFA1.s
000000r 1               ; 2) ld65 -C CFFA1.cfg  CFFA1.o -o CFFA1.bin
000000r 1               ;
000000r 1               ; Firmware Version History
000000r 1               ; ------------------------
000000r 1               ;
000000r 1               ; Version 1.0:
000000r 1               ;  Based on spdv6502 firmware from CFFA project
000000r 1               ;
000000r 1
000000r 1               ; For testing on an Apple II, use "-D APPLE2=1".
000000r 1               ; If the symbol is undefined, we set it to 0 here,
000000r 1               ; to target the Apple 1.
000000r 1                  .ifndef APPLE2
000000r 1               APPLE2              =               0
000000r 1                  .endif
000000r 1
000000r 1               DEBUG               =               0
000000r 1
000000r 1               .define             EQU             =
000000r 1               .define             TRUE            1
000000r 1               .define             FALSE           0
000000r 1
000000r 1               SPACE               EQU             $A0
000000r 1               CR                  EQU             $8D
000000r 1               ESC                 EQU             $9B
000000r 1
000000r 1               CTRL                EQU             $40
000000r 1
000000r 1               ; Apple1 mainboard I/O address definitions
000000r 1
000000r 1                                   .if APPLE2
000000r 1                                   .else
000000r 1               KEY_CONTROL         EQU             $D011
000000r 1               KEY_DATA            EQU             $D010
000000r 1               DSP_DATA            EQU             $D012
000000r 1               DSP_CONTROL         EQU             $D013
000000r 1               APPLE1_MON          EQU             $FF1F
000000r 1               BASIC_WARM          EQU             $E2B3
000000r 1                                   .endif
000000r 1
000000r 1               StackBase           EQU             $100
000000r 1
000000r 1               ;
000000r 1               ; Firmware Version Information
000000r 1               ;
000000r 1               FIRMWARE_VER        EQU             $01        ; Version of this code
000000r 1               OLDEST_COMPAT_VER   EQU             $01        ; oldest API version we are compatible with
000000r 1               BLOCKOFFSET         EQU               0        ; 0..255: LBA of first block of first partition
000000r 1               PARTITIONS32MB      EQU               4        ; Number of 32MB Partitions supported.
000000r 1
000000r 1               ;--------------------------------------------------------------------------------
000000r 1               ; Slot I/O definitions
000000r 1               ;
000000r 1                  .if APPLE2
000000r 1                  .else
000000r 1               IOBase             = $AFF0         ; Address is set by the Slot select line T, which is typically wired to $A000
```

```
4K select line from 74154
000000r 1
000000r 1               SetCSMask           = IOBase+1      ; Two special strobe locations to set and clear MASK bit that is used to
disable CS0 line to
000000r 1                                                   ; the CompactFlash during the CPU read cycles
000000r 1               ClearCSMask         = IOBase+2      ; that occur before every CPU write cycle. The normally innocuous read cycles
were causing the SanDisk CF to double increment
000000r 1                                                               ; during sector writes commands.
000000r 1
000000r 1               ATADevCtrl          = IOBase+6                  ; When writing
000000r 1               ATAAltStatus        = IOBase+6                  ; When reading
000000r 1               ATAData             = IOBase+8
000000r 1               ATAError            = IOBase+9                  ; When reading
000000r 1               ATAFeature          = IOBase+9                  ; When writing
000000r 1               ATASectorCnt        = IOBase+10
000000r 1               ATA_LBA07_00        = IOBase+11
000000r 1               ATA_LBA15_08        = IOBase+12
000000r 1               ATA_LBA23_16        = IOBase+13
000000r 1               ATA_LBA27_24        = IOBase+14
000000r 1               ATACommand          = IOBase+15                 ; When writing
000000r 1               ATAStatus           = IOBase+15                 ; When reading
000000r 1                  .endif
000000r 1
000000r 1               ;-------------------------------------------------------------------------
000000r 1               ; Zero-page RAM memory usage
000000r 1               PCL                 = $3A
000000r 1               PCH                 = $3B
000000r 1
000000r 1               LOMEM               = $4A
000000r 1               HIMEM               = $4C
000000r 1
000000r 1               ; ProDOS block interface locations
000000r 1               pdCommandCode       = $42
000000r 1               pdUnitNumber        = $43
000000r 1               pdIOBufferLow       = $44
000000r 1               pdIOBufferHigh      = $45
000000r 1               pdBlockNumberLow    = $46
000000r 1               pdBlockNumberHigh   = $47
000000r 1
000000r 1                  .if APPLE2
000000r 1               MiscZPStorage       = $00
000000r 1                  .else
000000r 1               MiscZPStorage       = $00  ; Currently approx 32 bytes used.
000000r 1                  .endif
000000r 1               ;
000000r 1               ; Inputs to the high-level routines (Rename, Delete, ReadFile, WriteFile, etc)
000000r 1               ;
000000r 1               Destination         = MiscZPStorage         ; 2 bytes
000000r 1               Filename            = Destination+2         ; 2 bytes
000000r 1               OldFilename         = Filename+2            ; 2 bytes
000000r 1               Filetype            = OldFilename+2
000000r 1               Auxtype             = Filetype+1            ; 2 bytes
000000r 1               FileSize            = Auxtype+2             ; 2 bytes
000000r 1               EntryPtr            = FileSize+2            ; 2 bytes
000000r 1               ;
000000r 1               ; internal use
000000r 1               ;
000000r 1               DirectoryBlock      = EntryPtr+2            ; 2 bytes - dir block # in Buffer
000000r 1               BitmapBase          = DirectoryBlock+2      ; 1 byte - low byte of block number (00..FF)
000000r 1               BitmapBlock         = BitmapBase+1          ; 1 byte - block currently in BitmapBuffer
000000r 1               BitmapDirty         = BitmapBlock+1
000000r 1               PastLastBitmapBlock = BitmapDirty+1         ; 1 byte - low byte of block number (00..FF)
000000r 1               TotalBlocks         = PastLastBitmapBlock+1 ; 2 bytes
000000r 1               UsedBlocks          = TotalBlocks+2         ; 2 bytes
000000r 1               FreeBlocks          = UsedBlocks+2          ; 2 bytes
000000r 1               EntryCounter        = FreeBlocks+2
000000r 1               BlockIndex          = EntryCounter+1        ; could easily move off of zero page
000000r 1               NameLen             = BlockIndex+1          ; could easily move off of zero page
000000r 1               LineCounter         = NameLen+1             ; could easily move off of zero page
000000r 1               digit_flag          = LineCounter+1         ; could easily move off of zero page
000000r 1               num                 = digit_flag+1          ; 2 bytes - could easily move off of zero page
000000r 1               ;
000000r 1               LastMiscZPPlusOne   = num+2
000000r 1
000000r 1               zpt1                = $F0          ;data at this location is saved/restored
000000r 1               MsgPointerLow       = $F2
000000r 1               MsgPointerHi        = $F3
000000r 1
000000r 1               InputBuffer         = $0200
000000r 1               InputBuffer2        = $0280
000000r 1
000000r 1               ;-------------------------------------------------------------------------
000000r 1               ; ProDOS directory entry structure
000000r 1               ;
000000r 1               oFiletype           = $10
000000r 1               oKeyBlock           = $11
000000r 1               oBlockCount         = $13
000000r 1               oFileSize           = $15
000000r 1               oCreateDateTime     = $18
000000r 1               oVersion            = $1C
000000r 1               oMinVersion         = $1D
000000r 1               oAccess             = $1E
000000r 1               oAuxtype            = $1F
000000r 1               oModDateTime        = $21
000000r 1               oHeaderPointer      = $25
000000r 1
000000r 1               kDirEntrySize       = $27
000000r 1               kEntriesPerBlock    = 13
000000r 1
```

```
000000r 1               oDirLinkPrevious    = $00
000000r 1               oDirLinkNext        = $02
000000r 1               oVolStorageType     = $04
000000r 1               oVolVersion         = $20
000000r 1               oVolAccess          = $22
000000r 1               oVolEntryLength     = $23
000000r 1               oVolEntriesPerBlock = $24
000000r 1               oVolFileCount       = $25
000000r 1               oVolBitmapNumber    = $27
000000r 1               oVolTotalBlocks     = $29
000000r 1
000000r 1               oSubdirParentPointer = $27
000000r 1               oSubdirParentEntryNum = $29
000000r 1               oSubdirParentEntryLength = $2A
000000r 1
000000r 1               kAccessDelete       = $80
000000r 1               kAccessRename       = $40
000000r 1               kAccessNeedsBackup  = $20
000000r 1               kAccessInvisible    = $04
000000r 1               kAccessWrite        = $02
000000r 1               kAccessRead         = $01
000000r 1               kAccessFull         = $C3
000000r 1
000000r 1               ; Storage types
000000r 1               kSeedling           = $10
000000r 1               kSapling            = $20
000000r 1               kTree               = $30
000000r 1               kExtended           = $50
000000r 1               kDirectory          = $D0
000000r 1               kSubdirHeader       = $E0
000000r 1               kVolume             = $F0
000000r 1               kStorageTypeMask    = $F0
000000r 1
000000r 1               ; Filetypes
000000r 1               kFiletypeText       = $04
000000r 1               kFiletypeBinary     = $06
000000r 1               kFiletypeDirectory  = $0F
000000r 1               kFiletypeBASIC1     = $F1
000000r 1               kFiletypeBAS        = $FC
000000r 1               kFiletypeSYS        = $FF
000000r 1
000000r 1               ; other constants
000000r 1               kRootDirectoryBlock = 2
000000r 1               kCanonicalFirstBitmapBlock = 6
000000r 1
000000r 1               ;-------------------------------------------------------------------------------
000000r 1
000000r 1               CFFA1_Initialized   = $8700       ; $A5 = we've initialized our globals
000000r 1               CFFA1_Options       = $8701       ; bit 7 = logging, bit 6 = terse menu
000000r 1               SpecialFirstBlock   = $8702       ; 0 = normal, else high byte of special first-block buffer
000000r 1               DriveNumber         = $8703       ; normally 0 to 3 for four 32MB partitions
000000r 1               DrvBlkCount0        = $8705       ; low byte of usable block count
000000r 1               DrvBlkCount1        = $8706       ; bits 8..15 of usable block count
000000r 1               DrvBlkCount2        = $8707       ; bits 16..23 of usable block count
000000r 1
000000r 1               API_A                   = $8710
000000r 1               StopAfter256FreeBlocks  = $8711
000000r 1
000000r 1               FirstDirectoryBlock     = $87AD  ; 2 bytes - block # of first block of current directory
000000r 1               ForceRootDirectorySearch = $87AF  ; bit 7 = OpenDir on main directory, ignoring PrefixDirectory
000000r 1               PrefixDirectory         = $87B0  ; 16 bytes (length + up to 15 characters)
000000r 1
000000r 1               CopyOfZeroPage          = $87C0  ; copy of MiscZPStorage
000000r 1                  .if LastMiscZPPlusOne-MiscZPStorage > $40
000000r 1                  .error "MiscZPStorage too large for CopyOfZeroPage"
000000r 1                  .endif
000000r 1               StagingBuffer       = $8800
000000r 1               buffer              = $8A00
000000r 1               DirectoryBuffer     = $8C00
000000r 1               BitmapBuffer        = $8E00
000000r 1
000000r 1               ;-------------------------------------------------------------------------------
000000r 1               ; Driver constant definitions
000000r 1               ;
000000r 1               ; ProDOS request Constants
000000r 1               PRODOS_STATUS       EQU             $00
000000r 1               PRODOS_READ         EQU             $01
000000r 1               PRODOS_WRITE        EQU             $02
000000r 1               PRODOS_FORMAT       EQU             $03
000000r 1
000000r 1               ; ProDOS low-level return codes
000000r 1               PRODOS_NO_ERROR     EQU             $00    ; No error
000000r 1               PRODOS_BADCMD       EQU             $01    ; Bad Command (not implemented)
000000r 1               PRODOS_IO_ERROR     EQU             $27    ; I/O error
000000r 1               PRODOS_NO_DEVICE    EQU             $28    ; No Device Connected
000000r 1               PRODOS_WRITE_PROTECT EQU            $2B    ; Write Protected
000000r 1               PRODOS_BADBLOCK     EQU             $2D    ; Invalid block number requested
000000r 1               PRODOS_OFFLINE      EQU             $2F    ; Device off-line
000000r 1               ; ProDOS high-level return codes
000000r 1               eBadPathSyntax      EQU             $40
000000r 1               eDirNotFound        EQU             $44
000000r 1               eFileNotFound       EQU             $46
000000r 1               eDuplicateFile      EQU             $47
000000r 1               eVolumeFull         EQU             $48
000000r 1               eDirectoryFull      EQU             $49
000000r 1               eFileFormat         EQU             $4A
000000r 1               eBadStrgType        EQU             $4B
000000r 1               eFileLocked         EQU             $4E
000000r 1               eNotProDOS          EQU             $52
```

```
000000r 1               eBadBufferAddr      EQU             $56
000000r 1               eBakedBitmap        EQU             $5A
000000r 1               eUnknownBASICFormat EQU             $FE
000000r 1               eUnimplemented      EQU             $FF
000000r 1
000000r 1               ; ATA Commands Codes
000000r 1               ATACRead            EQU             $20
000000r 1               ATACWrite           EQU             $30
000000r 1               ATAIdentify         EQU             $EC
000000r 1               ATASetFeature       EQU             $EF
000000r 1               Enable8BitTransfers EQU             $01
000000r 1
000000r 1               ; Constants for Wait
000000r 1               ;   Constant = (Delay[in uS]/2.5 + 2.09)^.5 - 2.7
000000r 1               WAIT_100ms          EQU             197
000000r 1               WAIT_40ms           EQU             124
000000r 1               WAIT_100us          EQU             4
000000r 1
000000r 1               .listbytes      unlimited
000000r 1
000000r 1               ;-------------------------------------------------------------------------------
000000r 1               ; CFFA1 firmware entry jump table
000000r 1               ;
000000r 1                .if APPLE2
000000r 1               Origin = $0A00                  ; CALL 2560
000000r 1                .else
000000r 1               Origin = $9000                  ; CALL -28672
000000r 1                .endif
000000r 1                  .ORG    Origin
009000  1
009000  1  4C D6 90        jmp     MenuExitToMonitor    ; $9000
009003  1  4C DC 90        jmp     MenuExitToBASIC      ; $9003
009006  1  4C E2 90        jmp     Menu                 ; $9006
009009  1  4C 98 A7        jmp     CFBlockDriver        ; $9009
00900C  1  4C 15 90        jmp     CFFA1_API            ; $900C
00900F  1               ; add jumps to new functionallity here....
00900F  1  EA              nop
009010  1  EA              nop
009011  1  EA              nop
009012  1  EA              nop
009013  1  EA              nop
009014  1  EA              nop
009015  1
009015  1               ;-------------------------------------------------------------------------------
009015  1               ; CFFA1_API -- Entry point for assembly programs to call CFFA1 ROM.
009015  1               ;
009015  1               ;    ldx #Command
009015  1               ;    jsr API
009015  1               ;
009015  1               ; Result:  CLC, A = 0 for success
009015  1               ;          SEC, A = error code
009015  1               CFFA1_API:
009015  1  E0 30           cpx #dispatchAPI_end-dispatchAPI
009017  1  B0 1A           bcs @badCommand
009019  1  8D 10 87        sta API_A
00901C  1  20 86 90        jsr InitializeGlobals
00901F  1  20 27 90        jsr @DispatchAPI
009022  1  B0 12           bcs @exit
009024  1  A9 00           lda #0
009026  1  60              rts
009027  1
009027  1               @DispatchAPI:
009027  1  BD 38 90        lda dispatchAPI+1,x
00902A  1  48              pha
00902B  1  BD 37 90        lda dispatchAPI,x
00902E  1  48              pha
00902F  1  AD 10 87        lda API_A
009032  1  60              rts
009033  1
009033  1               @badCommand:
009033  1  A9 01           lda #PRODOS_BADCMD
009035  1  38              sec
009036  1               @exit:
009036  1  60              rts
009037  1
009037  1               ;
009037  1               ; These API_* routines return with SEC/A=error, or with CLC (A=anything).
009037  1               ; The dispatcher takes care of setting A=0 in the no-error case.
009037  1               ;
009037  1               dispatchAPI:
009037  1  6A 90           .word API_Version-1             ; $00
009039  1  70 90           .word API_Menu-1                ; $02
00903B  1  75 90           .word API_DisplayError-1        ; $04
00903D  1  66 90           .word API_Reserved-1            ; $06
00903F  1  66 90           .word API_Reserved-1            ; $08
009041  1  66 90           .word API_Reserved-1            ; $0A
009043  1  66 90           .word API_Reserved-1            ; $0C
009045  1  66 90           .word API_Reserved-1            ; $0E
009047  1
009047  1  9A 9A           .word API_OpenDir-1             ; $10
009049  1  5C 9B           .word API_ReadDir-1             ; $12
00904B  1  F2 A5           .word API_FindDirEntry-1        ; $14
00904D  1  66 90           .word API_Reserved-1            ; $16
00904F  1  66 90           .word API_Reserved-1            ; $18
009051  1  66 90           .word API_Reserved-1            ; $1A
009053  1  66 90           .word API_Reserved-1            ; $1C
009055  1  66 90           .word API_Reserved-1            ; $1E
009057  1
009057  1  42 A2           .word API_WriteFile-1           ; $20
```

```
009059  1  55 9D           .word API_ReadFile-1            ; $22
00905B  1  85 A3           .word API_SaveBASICFile-1       ; $24
00905D  1  48 A3           .word API_LoadBASICFile-1       ; $26
00905F  1  0E A0           .word API_Rename-1              ; $28
009061  1  94 A0           .word API_Delete-1              ; $2A
009063  1  D1 A3           .word API_NewDirectoryAtRoot-1  ; $2C
009065  1  7A 90           .word API_FormatDrive-1         ; $2E
009067  1               dispatchAPI_end:
009067  1
009067  1               API_Reserved:
009067  1  A9 FF           lda #eUnimplemented
009069  1  38              sec
00906A  1  60              rts
00906B  1
00906B  1               API_Version:
00906B  1  A2 01           ldx #FIRMWARE_VER
00906D  1  A0 01           ldy #OLDEST_COMPAT_VER
00906F  1  18              clc
009070  1  60              rts
009071  1
009071  1               API_Menu:
009071  1  20 E2 90        jsr Menu
009074  1  18              clc
009075  1  60              rts
009076  1
009076  1               API_DisplayError:
009076  1  20 18 97        jsr DisplayError
009079  1  18              clc
00907A  1  60              rts
00907B  1
00907B  1               API_FormatDrive:
00907B  1  C0 77           cpy #$77          ; extra magic number to avoid accidental formatting
00907D  1  D0 03           bne @fail
00907F  1  4C A5 A4        jmp FormatDrive
009082  1               @fail:
009082  1  A9 01           lda #PRODOS_BADCMD
009084  1  38              sec
009085  1  60              rts
009086  1
009086  1               ;---------------------------------------------------------------------------
009086  1               ; InitializeGlobals
009086  1               ;
009086  1               ; Preserves:  X, Y.
009086  1               ;
009086  1               InitializeGlobals:
009086  1  A9 A5           lda #$A5
009088  1  CD 00 87        cmp CFFA1_Initialized
00908B  1  F0 11           beq @initialized
00908D  1  8D 00 87        sta CFFA1_Initialized
009090  1               ;
009090  1               ; Set up additional globals here.
009090  1               ;
009090  1  A9 00           lda #0
009092  1  8D 01 87        sta CFFA1_Options
009095  1  8D 03 87        sta DriveNumber
009098  1  8D B0 87        sta PrefixDirectory           ;empty prefix = use root of volume
00909B  1  8D AF 87        sta ForceRootDirectorySearch
00909E  1               @initialized:
00909E  1  60              rts
00909F  1
00909F  1               ;---------------------------------------------------------------------------
00909F  1               ;
00909F  1               ; Set up break handler vector for Apple1 ROM vector at: $100
00909F  1               ;
00909F  1               SetUpBreakHandler:
00909F  1                .if APPLE2
00909F  1                .else
00909F  1  A9 4C           lda   #$4C
0090A1  1  8D 00 01        sta   $100
0090A4  1  A9 AF           lda   #<BreakHandler
0090A6  1  8D 01 01        sta   $101
0090A9  1  A9 90           lda   #>BreakHandler
0090AB  1  8D 02 01        sta   $102
0090AE  1                .endif
0090AE  1  60              rts
0090AF  1
0090AF  1               ;---------------------------------------------------------------------------
0090AF  1               ;
0090AF  1               ; Break/IRQ handler
0090AF  1               ;
0090AF  1                .if APPLE2
0090AF  1                .else
0090AF  1               BreakHandler:
0090AF  1  68                   pla
0090B0  1  48                   pha             ;**irq handler
0090B1  1  0A                   asl
0090B2  1  0A                   asl
0090B3  1  0A                   asl
0090B4  1  30 0F                bmi   @Break      ;test for break
0090B6  1
0090B6  1               @IRQ:
0090B6  1  20 8C 96             jsr   DispString
0090B9  1  49 52 51 2E          .byte "IRQ...",CR,CR,0
0090BD  1  2E 2E 8D 8D
0090C1  1  00
0090C2  1  4C 1F FF             jmp   APPLE1_MON
0090C5  1
0090C5  1               @Break:
0090C5  1  20 8C 96             jsr   DispString
```

```
0090C8  1  42 52 45 41           .byte "BREAK...",CR,CR,0
0090CC  1  4B 2E 2E 2E
0090D0  1  8D 8D 00
0090D3  1  4C 1F FF              jmp   APPLE1_MON
0090D6  1                 .endif
0090D6  1
0090D6  1               ;--------------------------------------------------------------------------
0090D6  1               ;
0090D6  1               ; Run the menu, and then exit to the monitor.
0090D6  1               ;
0090D6  1               MenuExitToMonitor:
0090D6  1  20 E2 90        jsr     Menu
0090D9  1                 .if APPLE2
0090D9  1                   jmp     $FF69       ; Apple II monitor
0090D9  1                 .else
0090D9  1  4C 1F FF         jmp     APPLE1_MON
0090DC  1                 .endif
0090DC  1
0090DC  1               ;--------------------------------------------------------------------------
0090DC  1               ;
0090DC  1               ; Run the menu, and then exit to BASIC.
0090DC  1               ;
0090DC  1               MenuExitToBASIC:
0090DC  1  20 E2 90        jsr     Menu
0090DF  1                 .if APPLE2
0090DF  1                   jmp     $E003
0090DF  1                 .else
0090DF  1  4C B3 E2         jmp     BASIC_WARM
0090E2  1                 .endif
0090E2  1
0090E2  1
0090E2  1               ;--------------------------------------------------------------------------
0090E2  1               ; CFFA1 interactive menu
0090E2  1               ;
0090E2  1               Menu:
0090E2  1  20 40 91        jsr     SaveZeroPage
0090E5  1  20 86 90        jsr     InitializeGlobals
0090E8  1  20 9F 90        jsr     SetUpBreakHandler
0090EB  1               MenuAgain:
0090EB  1  20 4B 91        jsr     DisplayMenu
0090EE  1  20 53 98        jsr     WaitForKey              ; wait for user to press any key
0090F1  1  20 4A 98        jsr     DispChar                ; display char user typed
0090F4  1  20 8C 96        jsr     DispString
0090F7  1  8D 8D 00        .byte   CR,CR,0
0090FA  1
0090FA  1  29 7F           and     #$7F                    ; strip high bit before comparison
0090FC  1  20 81 96        jsr     UppercaseA
0090FF  1  A2 FC           ldx     #-4
009101  1               @FindMenuItem:
009101  1  E8              inx
009102  1  E8              inx
009103  1  E8              inx
009104  1  E8              inx
009105  1  BC 4E 92        ldy     MenuChoices,x
009108  1  F0 18           beq     UnknownMenuChoice
00910A  1  DD 4E 92        cmp     MenuChoices,x
00910D  1  D0 F2           bne     @FindMenuItem
00910F  1
00910F  1  BD 51 92        lda     MenuChoices+3,x
009112  1  85 01           sta     Destination+1
009114  1  BD 50 92        lda     MenuChoices+2,x
009117  1  85 00           sta     Destination
009119  1  20 1F 91        jsr     @DoMenuChoice
00911C  1  4C EB 90        jmp     MenuAgain
00911F  1
00911F  1               @DoMenuChoice:
00911F  1  6C 00 00        jmp     (Destination)
009122  1
009122  1               UnknownMenuChoice:
009122  1  20 8C 96        jsr     DispString
009125  1  3F 8D 00        .byte "?",CR,0
009128  1  2C 01 87        bit     CFFA1_Options
00912B  1  50 03           bvc     @1
00912D  1  20 60 91        jsr     DisplayOptions          ; in Terse mode, display options after invalid choice
009130  1               @1:
009130  1  4C EB 90        jmp     MenuAgain
009133  1
009133  1               MenuExit:
009133  1  68              pla                             ; discard the DoMenuChoice return address
009134  1  68              pla
009135  1               ;  v v v  Fall into  v v v
009135  1               ;--------------------------------------------------------------------------
009135  1               ; RestoreZeroPage
009135  1               ;
009135  1               RestoreZeroPage:
009135  1  A2 1F           ldx #LastMiscZPPlusOne-MiscZPStorage-1
009137  1               @restore:
009137  1  BD C0 87        lda CopyOfZeroPage,x
00913A  1  95 00           sta MiscZPStorage,x
00913C  1  CA              dex
00913D  1  10 F8           bpl @restore
00913F  1  60              rts
009140  1
009140  1               ;--------------------------------------------------------------------------
009140  1               ; SaveZeroPage
009140  1               ;
009140  1               SaveZeroPage:
009140  1  A2 1F           ldx #LastMiscZPPlusOne-MiscZPStorage-1
009142  1               @save:
```

```
009142  1  B5 00           lda MiscZPStorage,x
009144  1  9D C0 87        sta CopyOfZeroPage,x
009147  1  CA              dex
009148  1  10 F8           bpl @save
00914A  1  60              rts
00914B  1
00914B  1                ;--------------------------------------------------------------------------
00914B  1                DisplayMenu:
00914B  1  2C 01 87        bit CFFA1_Options
00914E  1  70 03           bvs @terse
009150  1  20 60 91        jsr DisplayOptions
009153  1                @terse:
009153  1  20 8C 96        jsr DispString
009156  1  8D 43 46 46     .byte CR,"CFFA1> ",0
00915A  1  41 31 3E 20
00915E  1  00
00915F  1  60              rts
009160  1
009160  1                DisplayOptions:
009160  1  20 8C 96        jsr DispString
009163  1  8D 20 43 46     .byte CR, " CFFA1 MENU (1.0)",CR
009167  1  46 41 31 20
00916B  1  4D 45 4E 55
00916F  1  20 28 31 2E
009173  1  30 29 8D
009176  1  20 2D 2D 2D     .byte     " ----------",CR
00917A  1  2D 2D 2D 2D
00917E  1  2D 2D 2D 8D
009182  1  20 43 20 2D     .byte " C - CATALOG      P - PREFIX",CR
009186  1  20 43 41 54
00918A  1  41 4C 4F 47
00918E  1  20 20 20 20
009192  1  20 20 50 20
009196  1  2D 20 50 52
00919A  1  45 46 49 58
00919E  1  8D
00919F  1  20 4C 20 2D     .byte " L - LOAD         N - NEW DIRECTORY",CR
0091A3  1  20 4C 4F 41
0091A7  1  44 20 20 20
0091AB  1  20 20 20 20
0091AF  1  20 20 4E 20
0091B3  1  2D 20 4E 45
0091B7  1  57 20 44 49
0091BB  1  52 45 43 54
0091BF  1  4F 52 59 8D
0091C3  1  20 53 20 2D     .byte " S - SAVE (BASIC) W - WRITE FILE",CR
0091C7  1  20 53 41 56
0091CB  1  45 20 28 42
0091CF  1  41 53 49 43
0091D3  1  29 20 57 20
0091D7  1  2D 20 57 52
0091DB  1  49 54 45 20
0091DF  1  46 49 4C 45
0091E3  1  8D
0091E4  1  20 52 20 2D     .byte " R - RENAME       D - DELETE",CR
0091E8  1  20 52 45 4E
0091EC  1  41 4D 45 20
0091F0  1  20 20 20 20
0091F4  1  20 20 44 20
0091F8  1  2D 20 44 45
0091FC  1  4C 45 54 45
009200  1  8D
009201  1  5E 46 20 2D     .byte "^F - FORMAT       T - TERSE",CR
009205  1  20 46 4F 52
009209  1  4D 41 54 20
00920D  1  20 20 20 20
009211  1  20 20 54 20
009215  1  2D 20 54 45
009219  1  52 53 45 8D
00921D  1  20 42 20 2D     .byte " B - READ BLOCK   M - MEMORY DISPLAY",CR
009221  1  20 52 45 41
009225  1  44 20 42 4C
009229  1  4F 43 4B 20
00922D  1  20 20 4D 20
009231  1  2D 20 4D 45
009235  1  4D 4F 52 59
009239  1  20 44 49 53
00923D  1  50 4C 41 59
009241  1  8D
009242  1                ;  .byte "^S - STATUS      ^D - DEBUG",CR
009242  1  20 51 20 2D     .byte " Q - QUIT",CR,0
009246  1  20 51 55 49
00924A  1  54 8D 00
00924D  1  60              rts
00924E  1
00924E  1                MenuChoices:
00924E  1  43 00 AF 98     .word 'C', MenuCatalog
009252  1  50 00 BB 94     .word 'P', MenuPrefix
009256  1  4C 00 C6 93     .word 'L', MenuLoadBASICOrBinaryFile
00925A  1  4E 00 9C 94     .word 'N', MenuNewDirectory
00925E  1  53 00 19 96     .word 'S', MenuSaveBASICFile
009262  1  57 00 2A 94     .word 'W', MenuWriteFile
009266  1  52 00 ED 94     .word 'R', MenuRenameFile
00926A  1  44 00 D5 94     .word 'D', MenuDeleteFile
00926E  1  06 00 61 95     .word 'F'-CTRL, MenuFormat
009272  1  54 00 36 95     .word 'T', MenuToggleTerse
009276  1  42 00 90 92     .word 'B', MenuReadBlock
00927A  1  4D 00 C5 92     .word 'M', MenuDisplayMemory
00927E  1  04 00 29 95     .word 'D'-CTRL, MenuToggleDebug
```

```
009282  1  13 00 64 93     .word 'S'-CTRL, MenuGetStatus
009286  1  51 00 33 91     .word 'Q', MenuExit
00928A  1                .if DEBUG
00928A  1                  .word 'Z'-CTRL, MenuWriteTwelveFiles
00928A  1                .endif
00928A  1  3F 00 60 91     .word '?', DisplayOptions
00928E  1  00 00           .word 0
009290  1
009290  1              ;--------------------------------------------------------------------------
009290  1              ; MenuReadBlock
009290  1              ;
009290  1              MenuReadBlock:
009290  1  20 8C 96        jsr DispString
009293  1  52 45 41 44     .byte "READ BLOCK: $",0
009297  1  20 42 4C 4F
00929B  1  43 4B 3A 20
00929F  1  24 00
0092A1  1  20 2F 96        jsr InputNumberAX
0092A4  1  90 01           bcc @notEmpty
0092A6  1  60              rts
0092A7  1              @notEmpty:
0092A7  1  20 19 A7        jsr UseBuffer
0092AA  1  20 90 A7        jsr ReadBlockAX
0092AD  1  B0 13           bcs @err
0092AF  1  A9 8A           lda #>buffer
0092B1  1  A2 00           ldx #<buffer
0092B3  1  85 01           sta Destination+1
0092B5  1  86 00           stx Destination
0092B7  1  A9 02           lda #>512
0092B9  1  A2 00           ldx #<512
0092BB  1  85 0A           sta FileSize+1
0092BD  1  86 09           stx FileSize
0092BF  1  4C FE 92        jmp DisplayMemory
0092C2  1              @err:
0092C2  1  4C 18 97        jmp DisplayError
0092C5  1
0092C5  1              ;--------------------------------------------------------------------------
0092C5  1              ; MenuDisplayMemory
0092C5  1              ;
0092C5  1              MenuDisplayMemory:
0092C5  1  20 8C 96        jsr DispString
0092C8  1  53 54 41 52     .byte "START: $",0
0092CC  1  54 3A 20 24
0092D0  1  00
0092D1  1  20 2F 96        jsr InputNumberAX
0092D4  1  B0 15           bcs @exit
0092D6  1  85 01           sta Destination+1
0092D8  1  86 00           stx Destination
0092DA  1
0092DA  1  20 8C 96        jsr DispString
0092DD  1  20 20 45 4E     .byte "  END: $",0
0092E1  1  44 3A 20 24
0092E5  1  00
0092E6  1  20 2F 96        jsr InputNumberAX
0092E9  1  90 01           bcc @go
0092EB  1              @exit:
0092EB  1  60              rts
0092EC  1              @go:
0092EC  1  A8              tay                ; compute FileSize = (AX - Destination) + 1
0092ED  1  8A              txa
0092EE  1  38              sec
0092EF  1  E5 00           sbc Destination
0092F1  1  AA              tax
0092F2  1  98              tya
0092F3  1  E5 01           sbc Destination+1
0092F5  1  A8              tay
0092F6  1  E8              inx
0092F7  1  D0 01           bne @1
0092F9  1  C8              iny
0092FA  1              @1:
0092FA  1  86 09           stx FileSize
0092FC  1  84 0A           sty FileSize+1
0092FE  1              ; v v v   fall into   v v v
0092FE  1              ;--------------------------------------------------------------------------
0092FE  1              ; DisplayMemory
0092FE  1              ;
0092FE  1              ; Input:  Destination (starting address), FileSize (number of bytes to display)
0092FE  1              ;
0092FE  1              DisplayMemory:
0092FE  1  20 5A 9A        jsr ResetLineCount
009301  1
009301  1  A5 09           lda FileSize
009303  1  D0 02           bne @noBorrow
009305  1  C6 0A           dec FileSize+1
009307  1              @noBorrow:
009307  1  C6 09           dec FileSize
009309  1
009309  1              @line:
009309  1  20 1F 93        jsr DisplayOneLine
00930C  1  B0 0E           bcs @exit
00930E  1  A5 09           lda FileSize
009310  1  E9 07           sbc #7            ; subtracts 8, since carry is clear
009312  1  85 09           sta FileSize
009314  1  A5 0A           lda FileSize+1
009316  1  E9 00           sbc #0
009318  1  85 0A           sta FileSize+1
00931A  1  B0 ED           bcs @line
00931C  1              @exit:
00931C  1  4C 43 98        jmp PressSpaceAndDispCR   ;SEC if we have already had ESC=abort
```

```
00931F  1
00931F  1               DisplayOneLine:
00931F  1  A5 01           lda Destination+1
009321  1  20 2A 98        jsr DispByte
009324  1  A5 00           lda Destination
009326  1  20 2A 98        jsr DispByte
009329  1  A9 BA           lda #':'+$80
00932B  1  20 4A 98        jsr DispChar
00932E  1  A0 00           ldy #0
009330  1               @byte:
009330  1  B1 00           lda (Destination),y
009332  1  20 2A 98        jsr DispByte
009335  1  20 3F 98        jsr DispSpace
009338  1  C8              iny
009339  1  C0 08           cpy #8
00933B  1  90 F3           bcc @byte
00933D  1
00933D  1  20 3F 98        jsr DispSpace
009340  1  A0 00           ldy #0
009342  1               @char:
009342  1  B1 00           lda (Destination),y
009344  1  20 59 93        jsr DisplayCharPrintable
009347  1  C8              iny
009348  1  C0 08           cpy #8
00934A  1  90 F6           bcc @char
00934C  1
00934C  1  98              tya
00934D  1  18              clc
00934E  1  65 00           adc Destination
009350  1  85 00           sta Destination
009352  1  90 02           bcc @samePage
009354  1  E6 01           inc Destination+1
009356  1               @samePage:
009356  1  4C 5F 9A        jmp PauseEveryNLines
009359  1
009359  1               DisplayCharPrintable:
009359  1  09 80           ora #$80
00935B  1  C9 A0           cmp #SPACE
00935D  1  B0 02           bcs @ok
00935F  1  A9 AE           lda #'.'+$80
009361  1               @ok:
009361  1  4C 4A 98        jmp DispChar
009364  1
009364  1
009364  1               ;--------------------------------------------------------------------------
009364  1               ; MenuGetStatus
009364  1               ;
009364  1               ; Prompt for drive number, call Status, and display the block count.
009364  1               ;
009364  1               MenuGetStatus:
009364  1  20 8C 96        jsr DispString
009367  1  44 52 49 56     .byte "DRIVE # (0-3): $",0
00936B  1  45 20 23 20
00936F  1  28 30 2D 33
009373  1  29 3A 20 24
009377  1  00
009378  1  20 2F 96        jsr InputNumberAX
00937B  1  B0 40           bcs @exit
00937D  1  8E 03 87        stx DriveNumber
009380  1
009380  1  20 8C 96        jsr DispString
009383  1  42 4C 4F 43     .byte "BLOCK COUNT FOR DRIVE ",0
009387  1  4B 20 43 4F
00938B  1  55 4E 54 20
00938F  1  46 4F 52 20
009393  1  44 52 49 56
009397  1  45 20 00
00939A  1  AD 03 87        lda DriveNumber
00939D  1  09 30           ora #'0'
00939F  1  20 4A 98        jsr DispChar
0093A2  1
0093A2  1  A9 00           lda #PRODOS_STATUS
0093A4  1  85 42           sta pdCommandCode
0093A6  1  20 98 A7        jsr CFBlockDriver
0093A9  1  B0 13           bcs @error
0093AB  1
0093AB  1  20 8C 96        jsr DispString
0093AE  1  20 3D 20 00     .byte " = ",0
0093B2  1  98              tya
0093B3  1  20 25 98        jsr DispByteWithDollarSign
0093B6  1  8A              txa
0093B7  1  20 2A 98        jsr DispByte
0093BA  1  20 48 98        jsr DispCR
0093BD  1
0093BD  1               @exit:
0093BD  1  60              rts
0093BE  1
0093BE  1               @error:
0093BE  1  20 8C 96        jsr DispString
0093C1  1  8D 00           .byte CR,0
0093C3  1  4C 18 97        jmp DisplayError
0093C6  1
0093C6  1               ;--------------------------------------------------------------------------
0093C6  1               ; MenuLoadBASICOrBinaryFile
0093C6  1               ;
0093C6  1               MenuLoadBASICOrBinaryFile:
0093C6  1  20 8C 96        jsr DispString
0093C9  1  20 20 20 4C     .byte "   LOAD FILE: ",0
0093CD  1  4F 41 44 20
```

```
0093D1  1  46 49 4C 45
0093D5  1  3A 20 00
0093D8  1  20 64 98        jsr GETLN
0093DB  1  F0 3D           beq @empty
0093DD  1
0093DD  1  A2 7F           ldx #127
0093DF  1                 @copyName:
0093DF  1  BD 00 02        lda InputBuffer,x
0093E2  1  9D 80 02        sta InputBuffer2,x
0093E5  1  CA              dex
0093E6  1  10 F7           bpl @copyName
0093E8  1
0093E8  1  A9 02           lda #>InputBuffer2
0093EA  1  A2 80           ldx #<InputBuffer2
0093EC  1  85 03           sta Filename+1
0093EE  1  86 02           stx Filename
0093F0  1  20 2C 9D        jsr GetFileInfo
0093F3  1  B0 2C           bcs @showError
0093F5  1
0093F5  1                 ; Is it a BASIC file?
0093F5  1  A5 06           lda Filetype
0093F7  1  C9 F1           cmp #$F1
0093F9  1  F0 29           beq @loadBASIC
0093FB  1
0093FB  1                 ; for any other file, prompt for load address (defaulting to file's Auxtype)
0093FB  1  20 8C 96        jsr DispString
0093FE  1  41 44 44 52     .byte "ADDR (",0
009402  1  20 28 00
009405  1  A5 08           lda Auxtype+1
009407  1  A6 07           ldx Auxtype
009409  1  20 15 9A        jsr PrintHexAX
00940C  1  20 8C 96        jsr DispString
00940F  1  29 3A 20 00     .byte "): ",0
009413  1  20 2F 96        jsr InputNumberAX
009416  1  F0 03           beq @default
009418  1  90 04           bcc @addrSupplied
00941A  1                 @error:
00941A  1                 @empty:
00941A  1  60              rts
00941B  1
00941B  1                 @default:
00941B  1  A9 00           lda #0
00941D  1  AA              tax
00941E  1                 @addrSupplied:
00941E  1  20 52 9D        jsr ReadFileAtAX
009421  1                 @showError:
009421  1  4C 18 97        jmp DisplayError
009424  1
009424  1                 @loadBASIC:
009424  1  20 49 A3        jsr LoadBASICFile
009427  1  4C 18 97        jmp DisplayError
00942A  1
00942A  1
00942A  1                  .if DEBUG
00942A  1                 ;------------------------------------------------------------------------------
00942A  1                 ; MenuWriteTwelveFiles
00942A  1                 ;
00942A  1                 MenuWriteTwelveFiles:
00942A  1
00942A  1                    lda #>InputBuffer
00942A  1                    ldx #<InputBuffer
00942A  1                    sta Filename+1
00942A  1                    stx Filename
00942A  1
00942A  1                    lda #1
00942A  1                    sta InputBuffer
00942A  1                    lda #'A'
00942A  1                    sta InputBuffer+1
00942A  1
00942A  1                 @nextFile:
00942A  1                    lda #>$1000
00942A  1                    ldx #<$1000
00942A  1                    sta Destination+1
00942A  1                    stx Destination
00942A  1                    sta Auxtype+1
00942A  1                    stx Auxtype
00942A  1
00942A  1                    lda #>$1234
00942A  1                    ldx #<$1234
00942A  1                    sta FileSize+1
00942A  1                    stx FileSize
00942A  1
00942A  1                    lda #kFiletypeBinary
00942A  1                    sta Filetype
00942A  1
00942A  1                    jsr WriteFile
00942A  1                    bcs @error
00942A  1
00942A  1                    jsr DispString
00942A  1                    .byte "Wrote file ",0
00942A  1                    lda InputBuffer+1
00942A  1                    jsr DispChar
00942A  1                    jsr DispCR
00942A  1
00942A  1                    inc InputBuffer+1
00942A  1                    lda InputBuffer+1
00942A  1                    cmp #'M'
00942A  1                    bcc @nextFile
00942A  1                    rts
```

```
00942A  1
00942A  1              @error:
00942A  1                 jmp DisplayError
00942A  1               .endif
00942A  1
00942A  1              ;------------------------------------------------------------------------
00942A  1              ; MenuWriteFile
00942A  1              ;
00942A  1              MenuWriteFile:
00942A  1  20 8C 96       jsr DispString
00942D  1  57 52 49 54    .byte "WRITE FROM: $",0
009431  1  45 20 46 52
009435  1  4F 4D 3A 20
009439  1  24 00
00943B  1  20 2F 96       jsr InputNumberAX
00943E  1  B0 5B          bcs @empty
009440  1  85 01          sta Destination+1
009442  1  86 00          stx Destination
009444  1  85 08          sta Auxtype+1
009446  1  86 07          stx Auxtype
009448  1
009448  1  20 8C 96       jsr DispString
00944B  1  20 20 20 20    .byte "    LENGTH: $",0
00944F  1  4C 45 4E 47
009453  1  54 48 3A 20
009457  1  24 00
009459  1  20 2F 96       jsr InputNumberAX
00945C  1  B0 3D          bcs @empty
00945E  1  85 0A          sta FileSize+1
009460  1  86 09          stx FileSize
009462  1
009462  1              ; Prompt for filetype
009462  1  20 8C 96       jsr DispString
009465  1  54 59 50 45    .byte "TYPE (BIN): $",0
009469  1  20 28 42 49
00946D  1  4E 29 3A 20
009471  1  24 00
009473  1  A9 06          lda #kFiletypeBinary
009475  1  85 06          sta Filetype
009477  1  20 2F 96       jsr InputNumberAX
00947A  1  F0 04          beq @defaultFiletype
00947C  1  B0 1D          bcs @error
00947E  1  86 06          stx Filetype
009480  1              @defaultFiletype:
009480  1
009480  1  20 8C 96       jsr DispString
009483  1  20 20 20 20    .byte "      NAME: ",0
009487  1  20 20 4E 41
00948B  1  4D 45 3A 20
00948F  1  00
009490  1  20 5C 98       jsr GetFilename
009493  1  F0 06          beq @empty
009495  1
009495  1  20 43 A2       jsr WriteFile
009498  1  20 18 97       jsr DisplayError
00949B  1
00949B  1              @empty:
00949B  1              @error:
00949B  1  60             rts
00949C  1
00949C  1              ;------------------------------------------------------------------------
00949C  1              ; MenuNewDirectory
00949C  1              ;
00949C  1              MenuNewDirectory:
00949C  1  20 8C 96       jsr DispString
00949F  1  4E 45 57 20    .byte "NEW DIRECTORY: ",0
0094A3  1  44 49 52 45
0094A7  1  43 54 4F 52
0094AB  1  59 3A 20 00
0094AF  1  20 5C 98       jsr GetFilename
0094B2  1  F0 06          beq @empty
0094B4  1
0094B4  1  20 D2 A3       jsr NewDirectoryAtRoot
0094B7  1  20 18 97       jsr DisplayError
0094BA  1              @empty:
0094BA  1  60             rts
0094BB  1
0094BB  1              ;------------------------------------------------------------------------
0094BB  1              ; MenuPrefix
0094BB  1              ;
0094BB  1              MenuPrefix:
0094BB  1  20 8C 96       jsr DispString
0094BE  1  50 52 45 46    .byte "PREFIX DIR: ",0
0094C2  1  49 58 20 44
0094C6  1  49 52 3A 20
0094CA  1  00
0094CB  1  20 5C 98       jsr GetFilename
0094CE  1  20 81 A4       jsr SetPrefix
0094D1  1  20 18 97       jsr DisplayError
0094D4  1  60             rts
0094D5  1
0094D5  1              ;------------------------------------------------------------------------
0094D5  1              ; MenuDeleteFile
0094D5  1              ;
0094D5  1              MenuDeleteFile:
0094D5  1  20 8C 96       jsr DispString
0094D8  1  44 45 4C 45    .byte "DELETE: ",0
0094DC  1  54 45 3A 20
0094E0  1  00
```

```
0094E1  1  20 5C 98        jsr GetFilename
0094E4  1  F0 06           beq @empty
0094E6  1
0094E6  1  20 95 A0        jsr Delete
0094E9  1  20 18 97        jsr DisplayError
0094EC  1               @empty:
0094EC  1  60              rts
0094ED  1
0094ED  1
0094ED  1               ;--------------------------------------------------------------------------
0094ED  1               ; MenuRenameFile
0094ED  1               ;
0094ED  1               MenuRenameFile:
0094ED  1  20 8C 96        jsr DispString
0094F0  1  52 45 4E 41     .byte "RENAME: ",0
0094F4  1  4D 45 3A 20
0094F8  1  00
0094F9  1  20 5C 98        jsr GetFilename
0094FC  1  F0 2A           beq @empty
0094FE  1
0094FE  1  A2 0F           ldx #15
009500  1               @copyOrigName:
009500  1  BD 00 02        lda InputBuffer,x
009503  1  9D 80 02        sta InputBuffer2,x
009506  1  CA              dex
009507  1  10 F7           bpl @copyOrigName
009509  1
009509  1  20 8C 96        jsr DispString
00950C  1  20 20 20 20     .byte "    TO: ",0
009510  1  54 4F 3A 20
009514  1  00
009515  1  20 5C 98        jsr GetFilename
009518  1  F0 0E           beq @empty
00951A  1
00951A  1               ; rename from InputBuffer2 to InputBuffer
00951A  1  A9 02           lda #>InputBuffer2
00951C  1  A2 80           ldx #<InputBuffer2
00951E  1  85 05           sta OldFilename+1
009520  1  86 04           stx OldFilename
009522  1  20 0F A0        jsr Rename
009525  1  20 18 97        jsr DisplayError
009528  1               @empty:
009528  1  60              rts
009529  1
009529  1
009529  1               ;--------------------------------------------------------------------------
009529  1               ; MenuToggleDebug - turn logging on/off
009529  1               ; MenuToggleTerse - turn the menu on/off
009529  1               ;
009529  1               MenuToggleDebug:
009529  1  20 8C 96        jsr DispString
00952C  1  44 45 42 55     .byte "DEBUG",0
009530  1  47 00
009532  1  A9 80           lda #$80
009534  1  D0 0B           bne toggleOptions
009536  1               MenuToggleTerse:
009536  1  20 8C 96        jsr DispString
009539  1  54 45 52 53     .byte "TERSE",0
00953D  1  45 00
00953F  1  A9 40           lda #$40
009541  1               toggleOptions:
009541  1  48              pha
009542  1  4D 01 87        eor CFFA1_Options   ; toggle the selected option
009545  1  8D 01 87        sta CFFA1_Options
009548  1  68              pla
009549  1  2D 01 87        and CFFA1_Options   ; check whether the toggled option is on or off
00954C  1  F0 09           beq @off
00954E  1  20 8C 96        jsr DispString
009551  1  20 4F 4E 8D     .byte " ON",CR,0
009555  1  00
009556  1  60              rts
009557  1               @off:
009557  1  20 8C 96        jsr DispString
00955A  1  20 4F 46 46     .byte " OFF",CR,0
00955E  1  8D 00
009560  1  60              rts
009561  1
009561  1               ;--------------------------------------------------------------------------
009561  1               ; MenuFormat
009561  1               ;
009561  1               MenuFormat:
009561  1  20 8C 96        jsr DispString
009564  1  44 45 53 54     .byte "DESTROY DATA ON DRIVE ",0
009568  1  52 4F 59 20
00956C  1  44 41 54 41
009570  1  20 4F 4E 20
009574  1  44 52 49 56
009578  1  45 20 00
00957B  1  AD 03 87        lda DriveNumber
00957E  1  09 30           ora #'0'
009580  1  20 4A 98        jsr DispChar
009583  1  20 8C 96        jsr DispString
009586  1  8D 8D 00        .byte CR,CR,0
009589  1
009589  1               ; If we're about to overwrite a ProDOS disk, present the name and the blocks-used
009589  1               ; to make sure the user knows what they're overwriting.
009589  1  20 C0 9B        jsr ComputeBlockCounts
00958C  1  B0 31           bcs @notProDOS
00958E  1
```

```
00958E  1  20 8C 96        jsr DispString
009591  1  44 45 53 54     .byte "DESTROY ",0
009595  1  52 4F 59 20
009599  1  00
00959A  1  20 1C 9A        jsr DisplayVolumeName
00959D  1
00959D  1  20 8C 96        jsr DispString
0095A0  1  8D 20 20 20     .byte CR,"   BLOCKS USED: ",0
0095A4  1  42 4C 4F 43
0095A8  1  4B 53 20 55
0095AC  1  53 45 44 3A
0095B0  1  20 00
0095B2  1  A5 16           lda UsedBlocks+1
0095B4  1  A6 15           ldx UsedBlocks
0095B6  1  20 CE 96        jsr PrintDecimalAX
0095B9  1  20 8C 96        jsr DispString
0095BC  1  8D 8D 00        .byte CR,CR,0
0095BF  1                  @notProDOS:
0095BF  1
0095BF  1                  ; Make sure they really want to erase.  Make them type YES Return.
0095BF  1  20 8C 96        jsr DispString
0095C2  1  41 52 45 20     .byte "ARE YOU SURE? ('YES') ",0
0095C6  1  59 4F 55 20
0095CA  1  53 55 52 45
0095CE  1  3F 20 28 27
0095D2  1  59 45 53 27
0095D6  1  29 20 00
0095D9  1  20 64 98        jsr GETLN
0095DC  1  F0 36           beq @exit
0095DE  1  A0 03           ldy #3
0095E0  1                  @checkYES:
0095E0  1  B9 00 02        lda InputBuffer,y
0095E3  1  D9 15 96        cmp @YES,Y
0095E6  1  D0 20           bne @aborted
0095E8  1  88              dey
0095E9  1  10 F5           bpl @checkYES
0095EB  1
0095EB  1                  ; Prompt for the new volume name and validate it.
0095EB  1  20 8C 96        jsr DispString
0095EE  1  8D 56 4F 4C     .byte CR,"VOLUME NAME: ",0
0095F2  1  55 4D 45 20
0095F6  1  4E 41 4D 45
0095FA  1  3A 20 00
0095FD  1  20 5C 98        jsr GetFilename
009600  1  F0 06           beq @aborted
009602  1  20 A5 A4        jsr FormatDrive
009605  1  4C 18 97        jmp DisplayError
009608  1
009608  1                  @aborted:
009608  1  20 8C 96        jsr DispString
00960B  1  41 42 4F 52     .byte "ABORTED",CR,0
00960F  1  54 45 44 8D
009613  1  00
009614  1                  @exit:
009614  1  60              rts
009615  1
009615  1                  @YES:
009615  1  03 59 45 53     .byte 3,"YES"
009619  1
009619  1                  ;--------------------------------------------------------------------------
009619  1                  MenuSaveBASICFile:
009619  1  20 8C 96        jsr DispString
00961C  1  53 41 56 45     .byte "SAVE: ",0
009620  1  3A 20 00
009623  1  20 5C 98        jsr GetFilename
009626  1  F0 06           beq @empty
009628  1  20 86 A3        jsr SaveBASICFile
00962B  1  4C 18 97        jmp DisplayError
00962E  1                  @empty:
00962E  1  60              rts
00962F  1
00962F  1                  ;--------------------------------------------------------------------------
00962F  1                  ; InputNumberAX
00962F  1                  ;
00962F  1                  ; Result:
00962F  1                  ;         BEQ if the user didn't enter anything at all
00962F  1                  ;
00962F  1                  ;         SEC if the user didn't enter a good number
00962F  1                  ;         CLC, AX = number
00962F  1                  ;
00962F  1                  ; Destroys InputBuffer.
00962F  1                  ;
00962F  1                  InputNumberAX:
00962F  1  20 64 98        jsr GETLN
009632  1  D0 02           bne @notEmpty
009634  1  38              sec
009635  1  60              rts
009636  1                  @notEmpty:
009636  1
009636  1                  ; parse the number (1 to 4 digits in hex)
009636  1  C9 05           cmp #5
009638  1  B0 19           bcs @exitNotEmpty
00963A  1
00963A  1  A0 00           ldy #0
00963C  1  84 1F           sty num+1
00963E  1  84 1E           sty num
009640  1                  @char:
009640  1  B9 01 02        lda InputBuffer+1,y
009643  1  20 56 96        jsr AccumulateHexDigit
```

```
009646  1  B0 0B          bcs @exitNotEmpty
009648  1  C8             iny
009649  1  CC 00 02       cpy InputBuffer
00964C  1  90 F2          bcc @char
00964E  1  A5 1F          lda num+1
009650  1  A6 1E          ldx num
009652  1  18             clc
009653  1              @exitNotEmpty:
009653  1  A0 01          ldy #1          ; BNE (not empty)
009655  1  60             rts
009656  1
009656  1              ;
009656  1              ; AccumulateHexDigit - Adds value of character A to Number.
009656  1              ;
009656  1              ; Result: CLC for success (Number adjusted)
009656  1              ;         SEC for error (not a hex character)
009656  1              ;
009656  1              ; Preserves Y.
009656  1              ;
009656  1              AccumulateHexDigit:
009656  1  A2 04          ldx #4
009658  1              @shift:
009658  1  06 1E          asl num
00965A  1  26 1F          rol num+1
00965C  1  CA             dex
00965D  1  D0 F9          bne @shift
00965F  1  20 81 96       jsr UppercaseA
009662  1  C9 47          cmp #'F'+1
009664  1  B0 19          bcs @digit_bad
009666  1  C9 41          cmp #'A'
009668  1  B0 0C          bcs @hexLetter
00966A  1  C9 3A          cmp #'9'+1
00966C  1  B0 11          bcs @digit_bad
00966E  1  C9 30          cmp #'0'
009670  1  90 0D          bcc @digit_bad
009672  1  29 0F          and #$0F
009674  1  10 03          bpl @addNumber   ; always taken
009676  1              @hexLetter:
009676  1  38             sec
009677  1  E9 37          sbc #'A'-10
009679  1              @addNumber:
009679  1  18             clc
00967A  1  65 1E          adc num
00967C  1  85 1E          sta num
00967E  1              ;  clc
00967E  1  60             rts
00967F  1
00967F  1              @digit_bad:
00967F  1  38             sec
009680  1  60             rts
009681  1
009681  1              UppercaseA:
009681  1  C9 7B          cmp #'z'+1
009683  1  B0 06          bcs @done
009685  1  C9 61          cmp #'a'
009687  1  90 02          bcc @done
009689  1  29 DF          and #%11011111
00968B  1              @done:
00968B  1  60             rts
00968C  1
00968C  1              ;-----------------------------------------------------------------------------
00968C  1              ; DispString - Sends a String to the Apple1's console
00968C  1              ; Input:
00968C  1              ;       string must immediately follow the JSR to this function
00968C  1              ;       and be terminated with zero byte.
00968C  1              ; Ouput:
00968C  1              ;       None
00968C  1              ;
00968C  1              ; ZeroPage Usage:
00968C  1              ;       MsgPointerLow, MsgPointerHi   (saved and restored)
00968C  1              ;
00968C  1              ; CPU Registers changed:  P
00968C  1              ;
00968C  1              ; Someday: Try compressing text by allowing $Fx = a run of blanks, $Ex = a run
00968C  1              ; of hyphens, and various other bytes to stand for commonly-used strings,
00968C  1              ; such as " BASIC ", "FILE", "BLOCK", "READ", "FROM", "NAME", " FULL", "BAD ".
00968C  1              ;
00968C  1              DispString:
00968C  1  48             pha                         ;save the Acc reg
00968D  1  8A             txa
00968E  1  48             pha                         ;save the X reg
00968F  1  98             tya
009690  1  48             pha                         ;save the Y reg
009691  1               .if APPLE2
009691  1                 jsr $fe89
009691  1                 jsr $fe93  ; disconnect BASIC.SYSTEM (which uses lots of zero page) from COUT and KEYIN
009691  1               .endif
009691  1  BA             tsx                         ;put the stack pointer in X
009692  1  A5 F3          lda  MsgPointerHi
009694  1  48             pha                         ;push zero page location on stack
009695  1  A5 F2          lda  MsgPointerLow
009697  1  48             pha                         ;push zero page location on stack
009698  1
009698  1  BD 04 01       lda  StackBase+4,x          ;determine the location of message to display
00969B  1  18             clc
00969C  1  69 01          adc  #$01                   ;add 1 because JSR pushes the last byte of its
00969E  1  85 F2          sta  MsgPointerLow          ; destination address on the stack
0096A0  1
0096A0  1  BD 05 01       lda  StackBase+5,x
```

```
0096A3  1  69 00           adc  #0
0096A5  1  85 F3           sta  MsgPointerHi
0096A7  1
0096A7  1              @dss1:
0096A7  1  A0 00           ldy  #0
0096A9  1  B1 F2           lda  (MsgPointerLow),y
0096AB  1  F0 0B           beq  @dssend
0096AD  1  20 4A 98        jsr  DispChar                ;display message
0096B0  1  E6 F2           inc  MsgPointerLow
0096B2  1  D0 F3           bne  @dss1
0096B4  1  E6 F3           inc  MsgPointerHi
0096B6  1  D0 EF           bne  @dss1
0096B8  1
0096B8  1              @dssend:
0096B8  1
0096B8  1  A5 F3           lda  MsgPointerHi
0096BA  1  9D 05 01        sta  StackBase+5,x
0096BD  1  A5 F2           lda  MsgPointerLow
0096BF  1  9D 04 01        sta  StackBase+4,x           ;fix up the return address on the stack.
0096C2  1
0096C2  1  68              pla
0096C3  1  85 F2           sta  MsgPointerLow           ;restore zero page location
0096C5  1  68              pla
0096C6  1  85 F3           sta  MsgPointerHi            ;restore zero page location
0096C8  1  68              pla
0096C9  1  A8              tay
0096CA  1  68              pla
0096CB  1  AA              tax
0096CC  1  68              pla
0096CD  1  60              rts                          ;return to location after string's null.
0096CE  1
0096CE  1
0096CE  1              ;--------------------------------------------------------------------------
0096CE  1              ; PrintDecimalAX
0096CE  1              ;
0096CE  1              ; Print a 16-bit unsigned number in decimal ("0" to "65535")
0096CE  1              ;
0096CE  1              PrintDecimalAX:
0096CE  1  46 1D           lsr digit_flag
0096D0  1  85 1F           sta num+1
0096D2  1  86 1E           stx num
0096D4  1
0096D4  1  A0 04           ldy #4
0096D6  1              @prd_l1:
0096D6  1  A2 B0           ldx #$80+'0' ;Digit so far
0096D8  1              @prd_l2:
0096D8  1  A5 1F           lda num+1
0096DA  1  D9 13 97        cmp @hi10,y
0096DD  1  D0 05           bne @pcmp_done
0096DF  1  A5 1E           lda num
0096E1  1  D9 0E 97        cmp @low10,y
0096E4  1              @pcmp_done:
0096E4  1  90 11           bcc @pr_digit
0096E6  1              ;  sec
0096E6  1  A5 1E           lda num
0096E8  1  F9 0E 97        sbc @low10,y
0096EB  1  85 1E           sta num
0096ED  1  A5 1F           lda num+1
0096EF  1  F9 13 97        sbc @hi10,y
0096F2  1  85 1F           sta num+1
0096F4  1  E8              inx
0096F5  1  D0 E1           bne @prd_l2
0096F7  1              @pr_digit:
0096F7  1  C0 00           cpy #0
0096F9  1  F0 08           beq @printit
0096FB  1  E0 B0           cpx #$80+'0'
0096FD  1  D0 04           bne @printit
0096FF  1  24 1D           bit digit_flag
009701  1  10 07           bpl @printed
009703  1              @printit:
009703  1  38              sec
009704  1  66 1D           ror digit_flag
009706  1              @printit2:
009706  1  8A              txa
009707  1  20 4A 98        jsr DispChar
00970A  1              @printed:
00970A  1  88              dey
00970B  1  10 C9           bpl @prd_l1
00970D  1  60              rts
00970E  1
00970E  1  01 0A 64 E8  @low10:  .byte $01,$0a,$64,$e8,$10
009712  1  10
009713  1  00 00 00 03  @hi10:   .byte $00,$00,$00,$03,$27
009717  1  27
009718  1
009718  1              ;--------------------------------------------------------------------------
009718  1              ; DisplayError
009718  1              ;
009718  1              ; Input: A
009718  1              ;
009718  1              ; Displays (CR) "xx [description]"
009718  1              ;
009718  1              DisplayError:
009718  1  20 8C 96        jsr DispString
00971B  1  8D 00           .byte CR,0
00971D  1
00971D  1  48              pha
00971E  1  20 2A 98        jsr DispByte
009721  1  68              pla
```

```
009722  1
009722  1  A2 1C          ldx #lastError-errorNumbers
009724  1               @search:
009724  1  DD 46 97       cmp errorNumbers,x
009727  1  F0 07          beq @errFound
009729  1  CA              dex
00972A  1  CA              dex
00972B  1  10 F7          bpl @search
00972D  1  4C 48 98       jmp DispCR
009730  1
009730  1               @errFound:
009730  1  BD 47 97       lda errorNumbers+1,x
009733  1  AA              tax
009734  1  20 3F 98       jsr DispSpace
009737  1  BD 64 97       lda sErrorStrings,x
00973A  1               @char:
00973A  1  20 4A 98       jsr DispChar
00973D  1  E8             inx
00973E  1  BD 64 97       lda sErrorStrings,x
009741  1  D0 F7          bne @char
009743  1  4C 48 98       jmp DispCR
009746  1
009746  1               errorNumbers:
009746  1  27 00          .byte PRODOS_IO_ERROR, sIOError-sErrorStrings
009748  1  2F 0A          .byte PRODOS_OFFLINE, sOffline-sErrorStrings
00974A  1  40 12          .byte eBadPathSyntax, sBadPathSyntax-sErrorStrings
00974C  1  46 1B          .byte eFileNotFound, sFileNotFound-sErrorStrings
00974E  1  47 2A          .byte eDuplicateFile, sDuplicateFile-sErrorStrings
009750  1  48 39          .byte eVolumeFull, sVolumeFull-sErrorStrings
009752  1  49 43          .byte eDirectoryFull, sDirectoryFull-sErrorStrings
009754  1  4B 52          .byte eBadStrgType, sBadStrgType-sErrorStrings
009756  1  4E 63          .byte eFileLocked, sFileLocked-sErrorStrings
009758  1  52 6A          .byte eNotProDOS, sNotProDOS-sErrorStrings
00975A  1  56 7A          .byte eBadBufferAddr, sBadBuffer-sErrorStrings
00975C  1  5A 8B          .byte eBakedBitmap, sBakedBitmap-sErrorStrings
00975E  1  FE 98          .byte eUnknownBASICFormat, sUnknownBASICFormat-sErrorStrings
009760  1  FF A9          .byte eUnimplemented, sUnimplemented-sErrorStrings
009762  1               lastError:
009762  1  00 B9          .byte 0, sSuccess-sErrorStrings
009764  1
009764  1               sErrorStrings:
009764  1               sIOError:
009764  1  49 2F 4F 20     .byte "I/O ERROR",0
009768  1  45 52 52 4F
00976C  1  52 00
00976E  1               sOffline:
00976E  1  4F 46 46 4C     .byte "OFFLINE",0
009772  1  49 4E 45 00
009776  1               sBadPathSyntax:
009776  1  42 41 44 20     .byte "BAD NAME",0
00977A  1  4E 41 4D 45
00977E  1  00
00977F  1               sFileNotFound:
00977F  1  46 49 4C 45     .byte "FILE NOT FOUND",0
009783  1  20 4E 4F 54
009787  1  20 46 4F 55
00978B  1  4E 44 00
00978E  1               sDuplicateFile:
00978E  1  44 55 50 4C     .byte "DUPLICATE FILE",0
009792  1  49 43 41 54
009796  1  45 20 46 49
00979A  1  4C 45 00
00979D  1               sVolumeFull:
00979D  1  44 49 53 4B     .byte "DISK FULL",0
0097A1  1  20 46 55 4C
0097A5  1  4C 00
0097A7  1               sDirectoryFull:
0097A7  1  44 49 52 45     .byte "DIRECTORY FULL",0
0097AB  1  43 54 4F 52
0097AF  1  59 20 46 55
0097B3  1  4C 4C 00
0097B6  1               sBadStrgType:
0097B6  1  42 41 44 20     .byte "BAD STORAGE TYPE",0
0097BA  1  53 54 4F 52
0097BE  1  41 47 45 20
0097C2  1  54 59 50 45
0097C6  1  00
0097C7  1               sFileLocked:
0097C7  1  4C 4F 43 4B     .byte "LOCKED",0
0097CB  1  45 44 00
0097CE  1               sNotProDOS:
0097CE  1  4E 4F 54 20     .byte "NOT PRODOS DISK",0
0097D2  1  50 52 4F 44
0097D6  1  4F 53 20 44
0097DA  1  49 53 4B 00
0097DE  1               sBadBuffer:
0097DE  1  42 41 44 20     .byte "BAD BUFF ADDRESS",0
0097E2  1  42 55 46 46
0097E6  1  20 41 44 44
0097EA  1  52 45 53 53
0097EE  1  00
0097EF  1               sBakedBitmap:
0097EF  1  42 41 4B 45     .byte "BAKED BITMAP",0
0097F3  1  44 20 42 49
0097F7  1  54 4D 41 50
0097FB  1  00
0097FC  1               sUnknownBASICFormat:
0097FC  1  4E 4F 54 20     .byte "NOT A BASIC FILE",0
009800  1  41 20 42 41
```

```
009804  1  53 49 43 20
009808  1  46 49 4C 45
00980C  1  00
00980D  1               sUnimplemented:
00980D  1  57 52 49 54     .byte "WRITE MORE CODE", 0
009811  1  45 20 4D 4F
009815  1  52 45 20 43
009819  1  4F 44 45 00
00981D  1               sSuccess:
00981D  1  53 55 43 43     .byte "SUCCESS",0
009821  1  45 53 53 00
009825  1               sPastStringTable:
009825  1                 .if sPastStringTable-sErrorStrings > 255
009825  1                 .error "ERROR MESSAGE TABLE TOO LARGE"
009825  1                 .endif
009825  1
009825  1               ;-----------------------------------------------------------------------------
009825  1               ; DispByte - Sends a Hex byte to the console
009825  1               ; Input:
009825  1               ;       A = Hex number to display
009825  1               ; Ouput:
009825  1               ;       None
009825  1               ;
009825  1               ; CPU Registers changed:  A, P
009825  1               ;
009825  1               DispByteWithDollarSign:
009825  1  20 8C 96        jsr DispString
009828  1  24 00           .byte "$",0
00982A  1               DispByte:
00982A  1  48              pha
00982B  1  4A              lsr a
00982C  1  4A              lsr a
00982D  1  4A              lsr a
00982E  1  4A              lsr a
00982F  1  20 33 98        jsr Nibble
009832  1  68              pla
009833  1               Nibble:
009833  1  29 0F           and #$0F
009835  1  09 30           ora #'0'
009837  1  C9 3A           cmp #'0'+10
009839  1  90 02           bcc @digit
00983B  1  69 06           adc #6
00983D  1               @digit:
00983D  1  D0 0B           bne DispChar   ; always taken
00983F  1
00983F  1               ;-----------------------------------------------------------------------------
00983F  1               ; DispCR, DispSpace, and DispChar - Sends a char to the Apple1 console
00983F  1               ;
00983F  1               ; Input:
00983F  1               ;       A = Character to Send
00983F  1               ; Ouput:
00983F  1               ;
00983F  1               ; ZeroPage Usage:
00983F  1               ;       None
00983F  1               ;
00983F  1               ; CPU Registers changed:  Acc, P
00983F  1               ;
00983F  1               ; X and Y are preserved.
00983F  1               ;
00983F  1               DispSpace:
00983F  1  A9 A0           lda #SPACE
009841  1  D0 07           bne DispChar
009843  1               PressSpaceAndDispCR:
009843  1  B0 03           bcs DispCR              ; already hit ESC to abort
009845  1  20 6E 9A        jsr PressSpaceOrESC
009848  1               DispCR:
009848  1  A9 8D           lda #CR
00984A  1               DispChar:
00984A  1                .if APPLE2
00984A  1                  ora      #$80
00984A  1                  stx      xsave
00984A  1                  sty      ysave
00984A  1                  jsr      $fded
00984A  1                  ldx      xsave
00984A  1                  ldy      ysave
00984A  1                  rts
00984A  1               xsave:
00984A  1                  .byte 0
00984A  1               ysave:
00984A  1                  .byte 0
00984A  1                .else
00984A  1               ; Wait until display is ready
00984A  1  2C 12 D0        bit      DSP_DATA
00984D  1  30 FB           bmi      DispChar
00984F  1               ; Send character to display hardware
00984F  1  8D 12 D0        sta      DSP_DATA
009852  1  60              rts
009853  1                .endif
009853  1
009853  1               ;-----------------------------------------------------------------------------
009853  1               ; WaitForKey - Wait until a key is pressed. Returns key stroke in Acc
009853  1               ; Input:
009853  1               ;      None
009853  1               ; Ouput:
009853  1               ;      Acc = ASCII value of key pressed
009853  1               ;
009853  1               ; X and Y are preserved.
009853  1               ;
009853  1               WaitForKey:
```

```
009853  1                .if APPLE2
009853  1                   stx wfk_xsave
009853  1                   sty wfk_ysave
009853  1                   jsr $fd0c
009853  1                   ldx wfk_xsave
009853  1                   ldy wfk_ysave
009853  1                   rts
009853  1              wfk_xsave:
009853  1                   .byte 0
009853  1              wfk_ysave:
009853  1                   .byte 0
009853  1                .else
009853  1  AD 11 D0         lda     KEY_CONTROL
009856  1  10 FB            bpl     WaitForKey
009858  1  AD 10 D0         lda     KEY_DATA
00985B  1  60               rts
00985C  1                .endif
00985C  1
00985C  1              ;------------------------------------------------------------------------------
00985C  1              ; GetFilename
00985C  1              ;
00985C  1              ; Result: Sets Filename = InputBuffer and calls GETLN.
00985C  1              ;         BEQ for an empty input
00985C  1              ;
00985C  1              GetFilename:
00985C  1  A9 02          lda #>InputBuffer
00985E  1  A2 00          ldx #<InputBuffer
009860  1  85 03          sta Filename+1
009862  1  86 02          stx Filename
009864  1              ;  v v v   FALL INTO   v v v
009864  1              ;------------------------------------------------------------------------------
009864  1              ; GETLN
009864  1              ;
009864  1              ; Read a string from the keyboard.
009864  1              ;
009864  1              ; Result:  InputBuffer (byte) = length of string
009864  1              ;          InputBuffer+1 = characters (high bit clear, terminated by a $00)
009864  1              ;          A = length
009864  1              ;          BEQ = empty string
009864  1              ;
009864  1              GETLN:
009864  1  A0 00          ldy #0
009866  1              getln1:
009866  1  20 53 98       jsr WaitForKey
009869  1  C9 8D          cmp #CR
00986B  1  F0 35          beq getln_done
00986D  1  C9 9B          cmp #ESC
00986F  1  F0 1B          beq getln_esc
009871  1  C9 98          cmp #$98              ; Ctrl-X cancel
009873  1  F0 17          beq getln_esc
009875  1  C9 DF          cmp #'_'+$80          ; Underscore = backspace
009877  1  F0 1E          beq getln_backspace
009879  1  C9 88          cmp #$88              ; Ctrl-H backspace
00987B  1  F0 1A          beq getln_backspace
00987D  1  C9 FF          cmp #$FF              ; Delete/Rubout
00987F  1  F0 14          beq getln_backspace0
009881  1  20 4A 98       jsr DispChar
009884  1  29 7F          and #$7f
009886  1  99 01 02       sta InputBuffer+1,y
009889  1  C8             iny
00988A  1  10 DA          bpl getln1
00988C  1              getln_esc:
00988C  1  20 8C 96       jsr DispString
00988F  1  5C 8D 00       .byte '\',CR,0
009892  1  4C 64 98       jmp GETLN
009895  1
009895  1              getln_backspace0:
009895  1  A9 88          lda #$88              ; print a Ctrl-H
009897  1              getln_backspace:
009897  1  C0 00          cpy #0
009899  1  F0 F1          beq getln_esc
00989B  1  20 4A 98       jsr DispChar
00989E  1  88             dey
00989F  1  4C 66 98       jmp getln1
0098A2  1
0098A2  1              getln_done:
0098A2  1  A9 00          lda #0
0098A4  1  99 01 02       sta InputBuffer+1,y
0098A7  1  8C 00 02       sty InputBuffer
0098AA  1  20 48 98       jsr DispCR
0098AD  1  98             tya
0098AE  1  60             rts
0098AF  1
0098AF  1
0098AF  1              ;------------------------------------------------------------------------------------------------
0098AF  1              ;------------------------------------------------------------------------------------------------
0098AF  1              ;------------------------------------------------------------------------------------------------
0098AF  1              ;------------------------------------------------------------------------------------------------
0098AF  1
0098AF  1
0098AF  1              ;------------------------------------------------------------------------------------------------
0098AF  1              ;
0098AF  1              ; Catalog
0098AF  1              ;
0098AF  1              ;  FILENAME        TYPE/AUX   BLOCKS
0098AF  1              ;  --------------- ---------  ------
0098AF  1              ; *FILENAME1234567 $TYP/$AUXT 12345
0098AF  1              ;
0098AF  1              ; (The "*" marks Locked items.)
```

```
0098AF  1               ;
0098AF  1               MenuCatalog:
0098AF  1  20 5A 9A        jsr ResetLineCount
0098B2  1  20 F1 9A        jsr OpenRootDir
0098B5  1  B0 3B           bcs @CatError
0098B7  1
0098B7  1  20 1C 9A        jsr DisplayVolumeName
0098BA  1  20 3B 9A        jsr DisplayPrefixDirectory
0098BD  1  20 5F 9A        jsr PauseEveryNLines
0098C0  1  20 43 99        jsr DisplayCatalogHeader
0098C3  1
0098C3  1               ; If there is a Prefix, call OpenDir (replacing our call to OpenRootDir above)
0098C3  1  AD B0 87        lda PrefixDirectory
0098C6  1  F0 05           beq @root
0098C8  1  20 9B 9A        jsr OpenDir
0098CB  1  B0 25           bcs @CatError
0098CD  1               @root:
0098CD  1
0098CD  1               @Cat1:
0098CD  1  20 5D 9B        jsr ReadDir
0098D0  1  B0 23           bcs @CatDone
0098D2  1  20 90 99        jsr CatLockedOrUnlocked
0098D5  1  20 A2 99        jsr CatFilename
0098D8  1  20 C5 99        jsr CatFiletype
0098DB  1  A9 2F           lda #'/'
0098DD  1  20 4A 98        jsr DispChar
0098E0  1  20 0D 9A        jsr CatAuxtype
0098E3  1  20 8C 96        jsr DispString
0098E6  1  20 20 00        .byte "  ",0
0098E9  1  20 02 9A        jsr CatBlockCount
0098EC  1  20 5F 9A        jsr PauseEveryNLines
0098EF  1  90 DC           bcc @Cat1
0098F1  1  60              rts
0098F2  1
0098F2  1               @CatError:
0098F2  1  4C 18 97        jmp DisplayError
0098F5  1
0098F5  1               ; Show the block counts:
0098F5  1               ;    BLKS FREE:nnnnn USED:nnnnn TOTAL:nnnnn
0098F5  1               @CatDone:
0098F5  1  C9 46           cmp #eFileNotFound
0098F7  1  D0 F9           bne @CatError
0098F9  1  20 C0 9B        jsr ComputeBlockCounts
0098FC  1  B0 F4           bcs @CatError
0098FE  1
0098FE  1  20 5F 9A        jsr PauseEveryNLines
009901  1  90 01           bcc @1
009903  1  60              rts
009904  1               @1:
009904  1  20 8C 96        jsr DispString
009907  1  42 4C 4B 53     .byte "BLKS FREE:",0
00990B  1  20 46 52 45
00990F  1  45 3A 00
009912  1  A5 18           lda FreeBlocks+1
009914  1  A6 17           ldx FreeBlocks
009916  1  20 CE 96        jsr PrintDecimalAX
009919  1
009919  1  20 8C 96        jsr DispString
00991C  1  20 55 53 45     .byte " USED:",0
009920  1  44 3A 00
009923  1  A5 16           lda UsedBlocks+1
009925  1  A6 15           ldx UsedBlocks
009927  1  20 CE 96        jsr PrintDecimalAX
00992A  1
00992A  1  20 8C 96        jsr DispString
00992D  1  20 54 4F 54     .byte " TOTAL:",0
009931  1  41 4C 3A 00
009935  1  A5 14           lda TotalBlocks+1
009937  1  A6 13           ldx TotalBlocks
009939  1  20 CE 96        jsr PrintDecimalAX
00993C  1  20 48 98        jsr DispCR
00993F  1  18              clc
009940  1  4C 43 98        jmp PressSpaceAndDispCR
009943  1
009943  1               DisplayCatalogHeader:
009943  1  20 8C 96        jsr DispString
009946  1  8D              .byte CR
009947  1  20 46 49 4C     .byte " FILENAME        TYPE/AUX   BLOCKS",CR
00994B  1  45 4E 41 4D
00994F  1  45 20 20 20
009953  1  20 20 20 20
009957  1  20 54 59 50
00995B  1  45 2F 41 55
00995F  1  58 20 20 20
009963  1  42 4C 4F 43
009967  1  4B 53 8D
00996A  1  20 2D 2D 2D     .byte " ---------------- ---------  ------",0
00996E  1  2D 2D 2D 2D
009972  1  2D 2D 2D 2D
009976  1  2D 2D 2D 2D
00997A  1  20 2D 2D 2D
00997E  1  2D 2D 2D 2D
009982  1  2D 2D 20 20
009986  1  2D 2D 2D 2D
00998A  1  2D 2D 00
00998D  1  4C 5F 9A        jmp PauseEveryNLines
009990  1
009990  1               ;
009990  1               ; Display a "*" for Locked or a blank for unlocked.
```

```
009990  1               ;
009990  1               CatLockedOrUnlocked:
009990  1  A0 1E           ldy #oAccess
009992  1  B1 0B           lda (EntryPtr),y
009994  1  29 C3           and #kAccessFull
009996  1  C9 C3           cmp #kAccessFull
009998  1  F0 05           beq unlocked
00999A  1  A9 2A           lda #'*'
00999C  1  4C 4A 98        jmp DispChar
00999F  1               unlocked:
00999F  1  4C 3F 98        jmp DispSpace
0099A2  1
0099A2  1               ;
0099A2  1               ; Display a filename (with trailing blanks)
0099A2  1               ;
0099A2  1               CatFilename:
0099A2  1  A0 00           ldy #0
0099A4  1  B1 0B           lda (EntryPtr),y
0099A6  1  29 0F           and #$0f
0099A8  1  A8              tay
0099A9  1  C8              iny
0099AA  1  84 1B           sty NameLen
0099AC  1  A0 01           ldy #1
0099AE  1               @char:
0099AE  1  B1 0B           lda (EntryPtr),y
0099B0  1  09 80           ora #$80
0099B2  1  20 4A 98        jsr DispChar
0099B5  1  C8              iny
0099B6  1  C4 1B           cpy NameLen
0099B8  1  90 F4           bcc @char
0099BA  1               @blanks:
0099BA  1  A9 20           lda #' '
0099BC  1  20 4A 98        jsr DispChar
0099BF  1  C8              iny
0099C0  1  C0 11           cpy #17
0099C2  1  90 F6           bcc @blanks
0099C4  1  60              rts
0099C5  1
0099C5  1               ;
0099C5  1               ; Display a filetype (three-letter abbreviation or $xx)
0099C5  1               ;
0099C5  1               CatFiletype:
0099C5  1  A0 10           ldy #oFiletype
0099C7  1  B1 0B           lda (EntryPtr),y
0099C9  1  A2 FC           ldx #-4
0099CB  1               @findFiletype:
0099CB  1  E8              inx
0099CC  1  E8              inx
0099CD  1  E8              inx
0099CE  1  E8              inx
0099CF  1  BC E9 99        ldy filetypeNames,x
0099D2  1  F0 12           beq @filetypeHex
0099D4  1  DD E9 99        cmp filetypeNames,x
0099D7  1  D0 F2           bne @findFiletype
0099D9  1
0099D9  1  A0 03           ldy #3
0099DB  1               @loop:
0099DB  1  BD EA 99        lda filetypeNames+1,x
0099DE  1  20 4A 98        jsr DispChar
0099E1  1  E8              inx
0099E2  1  88              dey
0099E3  1  D0 F6           bne @loop
0099E5  1  60              rts
0099E6  1
0099E6  1               @filetypeHex:
0099E6  1  4C 25 98        jmp DispByteWithDollarSign
0099E9  1
0099E9  1               filetypeNames:
0099E9  1  04 54 58 54     .byte $04,'T','X','T'
0099ED  1  06 42 49 4E     .byte $06,'B','I','N'
0099F1  1  0F 44 49 52     .byte $0F,'D','I','R'
0099F5  1  F1 42 41 31     .byte $F1,'B','A','1' ; "BA1" for Apple 1 BASIC
0099F9  1  FC 42 41 53     .byte $FC,'B','A','S'
0099FD  1  FF 53 59 53     .byte $FF,'S','Y','S'
009A01  1  00              .byte 0
009A02  1
009A02  1               ;
009A02  1               ; Display the block count (decimal, 0 to 65535)
009A02  1               ;
009A02  1               CatBlockCount:
009A02  1  A0 13           ldy #oBlockCount
009A04  1               CatDecimalWord:
009A04  1  B1 0B           lda (EntryPtr),y
009A06  1  AA              tax
009A07  1  C8              iny
009A08  1  B1 0B           lda (EntryPtr),y
009A0A  1  4C CE 96        jmp PrintDecimalAX
009A0D  1
009A0D  1               ;
009A0D  1               ; Display the auxtype ($xxxx)
009A0D  1               ;
009A0D  1               CatAuxtype:
009A0D  1  A0 1F           ldy #oAuxtype
009A0F  1               CatHexWord:
009A0F  1  B1 0B           lda (EntryPtr),y
009A11  1  AA              tax
009A12  1  C8              iny
009A13  1  B1 0B           lda (EntryPtr),y
009A15  1               PrintHexAX:
```

```
009A15  1  20 25 98        jsr DispByteWithDollarSign
009A18  1  8A               txa
009A19  1  4C 2A 98         jmp DispByte
009A1C  1
009A1C  1                ;------------------------------------------------------------------------------
009A1C  1                ; DisplayVolumeName
009A1C  1                ;
009A1C  1                ; Print "VOLUME: " and the name of the volume whose root directory block has
009A1C  1                ; been read into DirectoryBuffer.
009A1C  1                ;
009A1C  1                ; Destroys: A, X, Y.
009A1C  1                ;
009A1C  1                DisplayVolumeName:
009A1C  1  20 8C 96         jsr DispString
009A1F  1  56 4F 4C 55      .byte "VOLUME: ",0
009A23  1  4D 45 3A 20
009A27  1  00
009A28  1  AD 04 8C         lda DirectoryBuffer+oVolStorageType
009A2B  1  29 0F            and #$0f
009A2D  1  AA               tax
009A2E  1  A0 01            ldy #1
009A30  1                @volName:
009A30  1  B9 04 8C         lda DirectoryBuffer+oVolStorageType,y
009A33  1  20 4A 98         jsr DispChar
009A36  1  C8               iny
009A37  1  CA               dex
009A38  1  D0 F6            bne @volName
009A3A  1  60               rts
009A3B  1
009A3B  1                ;------------------------------------------------------------------------------
009A3B  1                ; DisplayPrefixDirectory
009A3B  1                ;
009A3B  1                ; Print "  DIR: " and the PrefixDirectory
009A3B  1                ;
009A3B  1                ; Destroys: A, X, Y.
009A3B  1                ;
009A3B  1                DisplayPrefixDirectory:
009A3B  1  AD B0 87         lda PrefixDirectory
009A3E  1  F0 19            beq @done
009A40  1  20 8C 96         jsr DispString
009A43  1  20 20 44 49      .byte "  DIR: ",0
009A47  1  52 3A 20 00
009A4B  1  A0 00            ldy #0
009A4D  1                @loop:
009A4D  1  B9 B1 87         lda PrefixDirectory+1,y
009A50  1  20 4A 98         jsr DispChar
009A53  1  C8               iny
009A54  1  CC B0 87         cpy PrefixDirectory
009A57  1  90 F4            bcc @loop
009A59  1                @done:
009A59  1  60               rts
009A5A  1
009A5A  1                ;------------------------------------------------------------------------------
009A5A  1                ; ResetLineCount
009A5A  1                ;
009A5A  1                ; Preserves the Carry flag.
009A5A  1                ;
009A5A  1                ResetLineCount:
009A5A  1  A9 14            lda #20
009A5C  1  85 1C            sta LineCounter
009A5E  1  60               rts
009A5F  1
009A5F  1                ;------------------------------------------------------------------------------
009A5F  1                ; PauseEveryNLines
009A5F  1                ;
009A5F  1                ; Result:  CLC to continue, SEC to abort
009A5F  1                ;
009A5F  1                ; X is preserved.
009A5F  1                ;
009A5F  1                PauseEveryNLines:
009A5F  1  20 48 98         jsr DispCR
009A62  1  18               clc
009A63  1  C6 1C            dec LineCounter
009A65  1  D0 06            bne @done
009A67  1  20 6E 9A         jsr PressSpaceOrESC
009A6A  1  20 5A 9A         jsr ResetLineCount
009A6D  1                @done:
009A6D  1  60               rts
009A6E  1
009A6E  1                ;------------------------------------------------------------------------------
009A6E  1                ; PressSpaceOrESC
009A6E  1                ;
009A6E  1                ; Result:  CLC to continue, SEC to abort
009A6E  1                ;
009A6E  1                ; X is preserved.
009A6E  1                ;
009A6E  1                PressSpaceOrESC:
009A6E  1  20 8C 96         jsr DispString
009A71  1  5B 20 53 50      .byte "[ SPACE/CR OR ESC ]",0
009A75  1  41 43 45 2F
009A79  1  43 52 20 4F
009A7D  1  52 20 45 53
009A81  1  43 20 5D 00
009A85  1                @key:
009A85  1  20 53 98         jsr WaitForKey
009A88  1  C9 9B            cmp #ESC
009A8A  1  F0 09            beq @exit           ; Carry set if taken
009A8C  1  C9 8D            cmp #CR
009A8E  1  F0 04            beq @space
```

```
009A90  1  C9 A0          cmp #SPACE
009A92  1  D0 F1          bne @key
009A94  1              @space:
009A94  1  18             clc
009A95  1              @exit:
009A95  1  08             php                ; preserve Carry
009A96  1  20 48 98       jsr DispCR
009A99  1  28             plp
009A9A  1  60             rts
009A9B  1
009A9B  1              ;------------------------------------------------------------------------------------------------
009A9B  1              ;------------------------------------------------------------------------------------------------
009A9B  1              ; API-level ProDOS disk access
009A9B  1              ;------------------------------------------------------------------------------------------------
009A9B  1              ;------------------------------------------------------------------------------------------------
009A9B  1
009A9B  1              ;--------------------------------------------------------------------------
009A9B  1              ; OpenDir
009A9B  1              ;
009A9B  1              ; Prepare for one or more ReadDir or ReadDir0 calls for PrefixDirectory.
009A9B  1              ; Output: CLC for success, SEC/A=error
009A9B  1              ;
009A9B  1              API_OpenDir:
009A9B  1              OpenDir:
009A9B  1  2C AF 87       bit ForceRootDirectorySearch
009A9E  1  30 51          bmi OpenRootDir
009AA0  1
009AA0  1  AD B0 87       lda PrefixDirectory    ; no prefix set?
009AA3  1  F0 4C          beq OpenRootDir
009AA5  1
009AA5  1              ; search for the Prefix directory, and prepare to iterate it instead
009AA5  1
009AA5  1  20 B0 9B       jsr DoNotUsePrefix
009AA8  1  A2 87          ldx #>PrefixDirectory
009AAA  1  A0 B0          ldy #<PrefixDirectory
009AAC  1  20 DD A5       jsr LocateDirEntryForNameXY    ; sets EntryPtr
009AAF  1  20 B7 9B       jsr UsePrefixNormally
009AB2  1  B0 24          bcs @error
009AB4  1
009AB4  1              ; EntryPtr points to our candidate directory
009AB4  1  20 D9 9A       jsr RequireDirectory
009AB7  1  B0 1F          bcs @error
009AB9  1
009AB9  1  20 A7 9B       jsr LoadKeyBlockIntoAX
009ABC  1  8D AE 87       sta FirstDirectoryBlock+1
009ABF  1  8E AD 87       stx FirstDirectoryBlock
009AC2  1  20 89 A7       jsr ReadDirectoryBlockAX
009AC5  1  B0 11          bcs @error
009AC7  1
009AC7  1              ; make sure this block looks like a valid subdirectory header
009AC7  1  AD 04 8C       lda DirectoryBuffer+4
009ACA  1  29 F0          and #kStorageTypeMask
009ACC  1  C9 E0          cmp #kSubdirHeader
009ACE  1  F0 04          beq @subdirOK
009AD0  1  A9 4B          lda #eBadStrgType
009AD2  1  38             sec
009AD3  1  60             rts
009AD4  1              @subdirOK:
009AD4  1  20 8E 9B       jsr InitEntryPtrAndCount
009AD7  1  18             clc
009AD8  1              @error:
009AD8  1  60             rts
009AD9  1
009AD9  1              ;--------------------------------------------------------------------------
009AD9  1              ; RequireDirectory
009AD9  1              ;
009AD9  1              ; Input:   EntryPtr points to a directory entry
009AD9  1              ;
009AD9  1              ; Output:  CLC for success, or SEC/A=error
009AD9  1              ;
009AD9  1              RequireDirectory:
009AD9  1  A0 00          ldy #0
009ADB  1  B1 0B          lda (EntryPtr),y
009ADD  1  29 F0          and #kStorageTypeMask
009ADF  1  C9 D0          cmp #kDirectory
009AE1  1  D0 0A          bne @notDirectory
009AE3  1  A0 10          ldy #oFiletype
009AE5  1  B1 0B          lda (EntryPtr),y
009AE7  1  C9 0F          cmp #kFiletypeDirectory
009AE9  1  D0 02          bne @notDirectory
009AEB  1  18             clc
009AEC  1  60             rts
009AED  1              @notDirectory:
009AED  1  A9 4B          lda #eBadStrgType
009AEF  1  38             sec
009AF0  1  60             rts
009AF1  1
009AF1  1              ;--------------------------------------------------------------------------
009AF1  1              ; OpenRootDir
009AF1  1              ;
009AF1  1              ; Prepare for one or more ReadDir or ReadDir0 calls for the root directory.
009AF1  1              ; Output: CLC for success, SEC/A=error
009AF1  1              ;
009AF1  1              OpenRootDir:
009AF1  1  20 8E 9B       jsr InitEntryPtrAndCount
009AF4  1              ReadMasterDirectoryBlock:
009AF4  1  A9 00          lda #>kRootDirectoryBlock
009AF6  1  A2 02          ldx #<kRootDirectoryBlock
009AF8  1  8D AE 87       sta FirstDirectoryBlock+1
```

```
009AFB  1  8E AD 87        stx FirstDirectoryBlock
009AFE  1  20 89 A7        jsr ReadDirectoryBlockAX
009B01  1  B0 59           bcs @exit
009B03  1
009B03  1  AD 04 8C        lda DirectoryBuffer+oVolStorageType
009B06  1  C9 F1           cmp #kVolume+1
009B08  1  90 4F           bcc @notProDOSFormat    ; $F1 to $FF is OK (volume with Name length 1 to 15)
009B0A  1
009B0A  1  AD 00 8C        lda DirectoryBuffer     ; link to previous block must be $0000
009B0D  1  0D 01 8C        ora DirectoryBuffer+1
009B10  1  D0 47           bne @notProDOSFormat
009B12  1
009B12  1  AD 23 8C        lda DirectoryBuffer+oVolEntryLength
009B15  1  C9 27           cmp #kDirEntrySize
009B17  1  D0 40           bne @notProDOSFormat
009B19  1  AD 24 8C        lda DirectoryBuffer+oVolEntriesPerBlock
009B1C  1  C9 0D           cmp #kEntriesPerBlock
009B1E  1  D0 39           bne @notProDOSFormat
009B20  1
009B20  1                ; require the bitmap base to be from 3 to 238, so all bitmap block numbers fit into 1 block
009B20  1  AD 28 8C           lda DirectoryBuffer+oVolBitmapNumber+1
009B23  1  D0 34              bne @notProDOSFormat
009B25  1  AE 27 8C           ldx DirectoryBuffer+oVolBitmapNumber
009B28  1  E0 03              cpx #3
009B2A  1  90 2D              bcc @notProDOSFormat
009B2C  1  E0 EF              cpx #239
009B2E  1  B0 29              bcs @notProDOSFormat
009B30  1  86 0F              stx BitmapBase
009B32  1
009B32  1  AD 2A 8C           lda DirectoryBuffer+oVolTotalBlocks+1
009B35  1  AE 29 8C           ldx DirectoryBuffer+oVolTotalBlocks
009B38  1  85 14              sta TotalBlocks+1
009B3A  1  86 13              stx TotalBlocks
009B3C  1  05 13              ora TotalBlocks
009B3E  1  F0 19              beq @notProDOSFormat
009B40  1
009B40  1                ; Compute number of bitmap blocks -- we have one for every 512*8 = 4096 ($1000)
009B40  1                ; blocks (or fraction thereof) on the volume.
009B40  1  A5 14              lda TotalBlocks+1
009B42  1  4A                 lsr a
009B43  1  4A                 lsr a
009B44  1  4A                 lsr a
009B45  1  4A                 lsr a
009B46  1  18                 clc
009B47  1  65 0F              adc BitmapBase
009B49  1  85 12              sta PastLastBitmapBlock
009B4B  1                ; if there is a fraction of $1000 blocks left over, we have one more bitmap block
009B4B  1  A5 14              lda TotalBlocks+1
009B4D  1  29 0F              and #$0F
009B4F  1  05 13              ora TotalBlocks
009B51  1  F0 02              beq @noFraction
009B53  1  E6 12              inc PastLastBitmapBlock
009B55  1                @noFraction:
009B55  1
009B55  1  A9 00              lda #0
009B57  1  18                 clc
009B58  1  60                 rts
009B59  1
009B59  1                @notProDOSFormat:
009B59  1  A9 52              lda #eNotProDOS
009B5B  1  38                 sec
009B5C  1                @exit:
009B5C  1  60                 rts
009B5D  1
009B5D  1                ;---------------------------------------------------------------------------
009B5D  1                ; ReadDir
009B5D  1                ;
009B5D  1                ; Find the next occupied directory entry.
009B5D  1                ; Result:  CLC for success, EntryPtr is set, DirectoryBlock is set, A=$s0 (storage type)
009B5D  1                ;          SEC/A=error
009B5D  1                ;
009B5D  1                API_ReadDir:
009B5D  1                ReadDir:
009B5D  1  20 65 9B           jsr ReadDir0
009B60  1  B0 02              bcs @err
009B62  1  F0 F9              beq ReadDir
009B64  1                @err:
009B64  1  60                 rts
009B65  1
009B65  1                ;---------------------------------------------------------------------------
009B65  1                ; ReadDir0
009B65  1                ;
009B65  1                ; Find the next directory entry, whether occupied or not.
009B65  1                ;
009B65  1                ; Result:  CLC for success
009B65  1                ;              EntryPtr is set
009B65  1                ;              DirectoryBlock is set,
009B65  1                ;              A=$s0 (storage type)
009B65  1                ;              BEQ for an unused entry
009B65  1                ;          SEC/A=error
009B65  1                ;
009B65  1                ReadDir0:
009B65  1  20 9B 9B           jsr BumpEntryPtr
009B68  1  C6 19              dec EntryCounter
009B6A  1  D0 16              bne @sameBlock
009B6C  1
009B6C  1  AD 03 8C           lda DirectoryBuffer+oDirLinkNext+1
009B6F  1  0D 02 8C           ora DirectoryBuffer+oDirLinkNext
009B72  1  F0 16              beq @noMore
```

```
009B74  1  AD 03 8C        lda DirectoryBuffer+oDirLinkNext+1
009B77  1  AE 02 8C        ldx DirectoryBuffer+oDirLinkNext
009B7A  1  20 89 A7        jsr ReadDirectoryBlockAX
009B7D  1  B0 0E           bcs @done
009B7F  1  20 8E 9B        jsr InitEntryPtrAndCount
009B82  1                 @sameBlock:
009B82  1  A0 00           ldy #0
009B84  1  B1 0B           lda (EntryPtr),y
009B86  1  29 F0           and #kStorageTypeMask     ; storage type in A ($s0), BEQ = 0
009B88  1  18              clc
009B89  1  60              rts
009B8A  1                 @noMore:
009B8A  1  A9 46           lda #eFileNotFound
009B8C  1  38              sec
009B8D  1                 @done:
009B8D  1  60              rts
009B8E  1
009B8E  1
009B8E  1                 ;------------------------------------------------------------------------------
009B8E  1                 ; InitEntryPtr
009B8E  1                 ;
009B8E  1                 InitEntryPtrAndCount:
009B8E  1  A9 0D           lda #kEntriesPerBlock
009B90  1  85 19           sta EntryCounter
009B92  1                 InitEntryPtr:
009B92  1  A9 8C           lda #>DirectoryBuffer
009B94  1  A2 04           ldx #<DirectoryBuffer+4
009B96  1  85 0C           sta EntryPtr+1
009B98  1  86 0B           stx EntryPtr
009B9A  1  60              rts
009B9B  1
009B9B  1                 ;------------------------------------------------------------------------------
009B9B  1                 ; BumpEntryPtr
009B9B  1                 ;
009B9B  1                 BumpEntryPtr:
009B9B  1  18              clc
009B9C  1  A5 0B           lda EntryPtr
009B9E  1  69 27           adc #kDirEntrySize
009BA0  1  85 0B           sta EntryPtr
009BA2  1  90 02           bcc @noCarry
009BA4  1  E6 0C           inc EntryPtr+1
009BA6  1                 @noCarry:
009BA6  1  60              rts
009BA7  1
009BA7  1                 ;------------------------------------------------------------------------------
009BA7  1                 ; LoadKeyBlockIntoAX
009BA7  1                 ;
009BA7  1                 ; Input:    EntryPtr
009BA7  1                 ; Output:   A, X (from oKeyBlock field of EntryPtr)
009BA7  1                 ; Destroys: Y, P
009BA7  1                 ;
009BA7  1                 LoadKeyBlockIntoAX:
009BA7  1  A0 11           ldy #oKeyBlock
009BA9  1  B1 0B           lda (EntryPtr),y
009BAB  1  AA              tax
009BAC  1  C8              iny
009BAD  1  B1 0B           lda (EntryPtr),y
009BAF  1  60              rts
009BB0  1
009BB0  1
009BB0  1                 ;------------------------------------------------------------------------------
009BB0  1                 ; DoNotUsePrefix
009BB0  1                 ;
009BB0  1                 ; Future calls to OpenDir will operate in the root directory, not PrefixDirectory.
009BB0  1                 ;
009BB0  1                 ; Preserves all registers (even P).
009BB0  1                 ;
009BB0  1                 DoNotUsePrefix:
009BB0  1  08              php
009BB1  1  38              sec
009BB2  1  6E AF 87        ror ForceRootDirectorySearch
009BB5  1  28              plp
009BB6  1  60              rts
009BB7  1
009BB7  1                 ;------------------------------------------------------------------------------
009BB7  1                 ; UsePrefixNormally
009BB7  1                 ;
009BB7  1                 ; Future calls to OpenDir will operate in the PrefixDirectory.
009BB7  1                 ;
009BB7  1                 ; Preserves all registers (even P).
009BB7  1                 ;
009BB7  1                 UsePrefixNormally:
009BB7  1  08              php
009BB8  1  4E AF 87        lsr ForceRootDirectorySearch
009BBB  1  28              plp
009BBC  1  60              rts
009BBD  1
009BBD  1
009BBD  1                 ;------------------------------------------------------------------------------
009BBD  1                 ; ComputeBlockCounts
009BBD  1                 ;
009BBD  1                 ; Result:  SEC/A=error
009BBD  1                 ;          CLC, TotalBlocks, FreeBlocks, UsedBlocks
009BBD  1                 ;
009BBD  1                 ComputeFreeBlocksUpTo256:
009BBD  1  38              sec
009BBE  1  B0 01           bcs computeCommon
009BC0  1                 ComputeBlockCounts:
009BC0  1  18              clc
```

```
009BC1  1              computeCommon:
009BC1  1  6E 11 87       ror StopAfter256FreeBlocks
009BC4  1
009BC4  1  20 F4 9A       jsr ReadMasterDirectoryBlock
009BC7  1  B0 2F          bcs @exit
009BC9  1
009BC9  1  20 27 9C       jsr ReadFirstBitmapBlock
009BCC  1  B0 2A          bcs @exit
009BCE  1
009BCE  1              ; count the free blocks (bits set to "1") in every bitmap block
009BCE  1  A9 00          lda #0
009BD0  1  85 18          sta FreeBlocks+1
009BD2  1  85 17          sta FreeBlocks
009BD4  1
009BD4  1              @CountOneBlock:
009BD4  1  A0 00          ldy #0
009BD6  1              @loop:
009BD6  1  B9 00 8E       lda BitmapBuffer,y
009BD9  1  20 08 9C       jsr AddFreeBlocksFromA
009BDC  1  B9 00 8F       lda BitmapBuffer+256,y
009BDF  1  20 08 9C       jsr AddFreeBlocksFromA
009BE2  1  C8             iny
009BE3  1  D0 F1          bne @loop
009BE5  1
009BE5  1  AD 11 87       lda StopAfter256FreeBlocks
009BE8  1  10 04          bpl @countAll
009BEA  1  A5 18          lda FreeBlocks+1
009BEC  1  D0 0B          bne @done
009BEE  1              @countAll:
009BEE  1
009BEE  1  20 4C 9C       jsr ReadNextBitmapBlock
009BF1  1  90 E1          bcc @CountOneBlock
009BF3  1  C9 46          cmp #eFileNotFound
009BF5  1  F0 02          beq @done
009BF7  1  38             sec
009BF8  1              @exit:
009BF8  1  60             rts
009BF9  1
009BF9  1              @done:
009BF9  1              ; compute Used blocks = Total - Free
009BF9  1  38             sec
009BFA  1  A5 13          lda TotalBlocks
009BFC  1  E5 17          sbc FreeBlocks
009BFE  1  85 15          sta UsedBlocks
009C00  1  A5 14          lda TotalBlocks+1
009C02  1  E5 18          sbc FreeBlocks+1
009C04  1  85 16          sta UsedBlocks+1
009C06  1  18             clc
009C07  1  60             rts
009C08  1
009C08  1              ;--------------------------------------------------------------------------
009C08  1              ; AddFreeBlocksFromA
009C08  1              ;
009C08  1              ; Input:  A = a byte from a bitmap block
009C08  1              ; Result: increment FreeBlocks by the number of "1" bits set in A
009C08  1              ;
009C08  1              ; Destroys: A, X.
009C08  1              ; Preserves: Y.
009C08  1              ;
009C08  1              AddFreeBlocksFromA:
009C08  1  A2 08          ldx #8         ; anticipate 8 free blocks
009C0A  1  C9 FF          cmp #$ff
009C0C  1  F0 0C          beq @done      ; special case for 8 free blocks, since it happens often
009C0E  1
009C0E  1  A2 00          ldx #0         ; count up the "1" bits one at a time
009C10  1              @loop:
009C10  1  C9 00          cmp #0
009C12  1  F0 06          beq @done
009C14  1  4A             lsr a
009C15  1  90 F9          bcc @loop
009C17  1  E8             inx
009C18  1  D0 F6          bne @loop
009C1A  1              @done:
009C1A  1  8A             txa
009C1B  1  F0 09          beq @exit
009C1D  1  18             clc
009C1E  1  65 17          adc FreeBlocks
009C20  1  85 17          sta FreeBlocks
009C22  1  90 02          bcc @exit
009C24  1  E6 18          inc FreeBlocks+1
009C26  1              @exit:
009C26  1  60             rts
009C27  1
009C27  1              ;--------------------------------------------------------------------------
009C27  1              ; ReadFirstBitmapBlock, ReadBitmapBlockOffsetA
009C27  1              ;
009C27  1              ; Assumes that ReadMasterDirectoryBlock has already been called.
009C27  1              ;
009C27  1              ; Result:  SEC/A=error
009C27  1              ;          CLC, BitmapBlock set, BitmapBuffer filled
009C27  1              ;
009C27  1              ReadFirstBitmapBlock:
009C27  1  A9 00          lda #0
009C29  1  85 10          sta BitmapBlock         ; means no bitmap block has already been read in
009C2B  1  85 11          sta BitmapDirty
009C2D  1              ReadBitmapBlockOffsetA:
009C2D  1  18             clc
009C2E  1  65 0F          adc BitmapBase
009C30  1              ReadBitmapBlockA:
```

```
009C30  1  C5 10          cmp BitmapBlock
009C32  1  F0 12          beq @sameBlock
009C34  1  48             pha
009C35  1  20 1A 9D       jsr FlushBitmap
009C38  1  B0 0E          bcs @error1
009C3A  1  68             pla
009C3B  1  AA             tax
009C3C  1  86 10          stx BitmapBlock
009C3E  1  A9 00          lda #0
009C40  1  20 21 A7       jsr UseBitmapBuffer
009C43  1  4C 90 A7       jmp ReadBlockAX
009C46  1                 @sameBlock:
009C46  1  18             clc
009C47  1  60             rts
009C48  1                 @error1:
009C48  1  AA             tax
009C49  1  68             pla
009C4A  1  8A             txa
009C4B  1  60             rts
009C4C  1
009C4C  1                 ;------------------------------------------------------------------------------
009C4C  1                 ; ReadNextBitmapBlock
009C4C  1                 ;
009C4C  1                 ; Result: SEC/A=error
009C4C  1                 ;         SEC/A=eFileNotFound if there are no more blocks
009C4C  1                 ;         CLC, BitmapBuffer filled in, BitmapBlock adjusted
009C4C  1                 ;
009C4C  1                 ReadNextBitmapBlock:
009C4C  1  20 1A 9D       jsr FlushBitmap
009C4F  1  B0 0B          bcs @exit
009C51  1  A6 10          ldx BitmapBlock
009C53  1  E8             inx
009C54  1  8A             txa
009C55  1  C5 12          cmp PastLastBitmapBlock
009C57  1  90 D7          bcc ReadBitmapBlockA
009C59  1  A9 46          lda #eFileNotFound
009C5B  1  38             sec
009C5C  1                 @exit:
009C5C  1  60             rts
009C5D  1
009C5D  1                 ;------------------------------------------------------------------------------
009C5D  1                 ; AllocateOneBlockAX
009C5D  1                 ;
009C5D  1                 ; Result:  CLC, AX = block number of a freshly allocated block
009C5D  1                 ;               Block number is also stored in pdBlockNumber.
009C5D  1                 ;               BitmapDirty is set, BitmapBuffer is updated.
009C5D  1                 ;          SEC, A = error
009C5D  1                 ;
009C5D  1                 ; You must have previously called ReadFirstBitmapBlock.
009C5D  1                 ; Later, you have to call FlushBitmap.
009C5D  1                 ;
009C5D  1                 AllocateOneBlockAX:
009C5D  1  18             clc
009C5E  1  A0 00          ldy #0
009C60  1                 @scan:
009C60  1  B9 00 8E       lda BitmapBuffer,y
009C63  1  D0 12          bne @found_free
009C65  1  C8             iny
009C66  1  D0 F8          bne @scan
009C68  1
009C68  1  38             sec
009C69  1                 @scan2:
009C69  1  B9 00 8F       lda BitmapBuffer+256,y
009C6C  1  D0 09          bne @found_free
009C6E  1  C8             iny
009C6F  1  D0 F8          bne @scan2
009C71  1
009C71  1  20 4C 9C       jsr ReadNextBitmapBlock
009C74  1  90 E7          bcc AllocateOneBlockAX
009C76  1  60             rts
009C77  1
009C77  1                 ; SEC indicates the free block was in the 2nd half of the bitmap block
009C77  1                 @found_free:
009C77  1  08             php
009C78  1  48             pha
009C79  1  A2 FF          ldx #-1
009C7B  1  86 11          stx BitmapDirty
009C7D  1                 @bit:
009C7D  1  E8             inx
009C7E  1  0A             asl a
009C7F  1  90 FC          bcc @bit
009C81  1  68             pla
009C82  1  28             plp
009C83  1                 ; X is now 0..7, the number of the lowest free block (1 bit) in the group of 8
009C83  1                 ; A is the byte, and Y is its offset within half the block, and SEC = 2nd half
009C83  1  3D BF 9C       and BitmapClearMasks,x
009C86  1  B0 05          bcs @secondHalf
009C88  1  99 00 8E       sta BitmapBuffer,y
009C8B  1  90 03          bcc @more
009C8D  1                 @secondHalf:
009C8D  1  99 00 8F       sta BitmapBuffer+256,y
009C90  1                 @more:
009C90  1                 ;
009C90  1                 ; BlockNumber = (4096 * blockoffset) + (8 * Y) + (2048 * 2nd-half) + bit-position
009C90  1                 ;     BlockNumberHigh = 16*blockoffset + 8*2nd-half + Y/32
009C90  1                 ;     BlockNumberLow = (Y << 3) + bit-position
009C90  1                 ;
009C90  1  A9 08          lda #8
009C92  1  B0 02          bcs @plus2048
```

```
009C94  1  A9 00          lda #0
009C96  1               @plus2048:
009C96  1  85 47           sta pdBlockNumberHigh
009C98  1  98              tya
009C99  1  4A              lsr a
009C9A  1  4A              lsr a
009C9B  1  4A              lsr a
009C9C  1  4A              lsr a
009C9D  1  4A              lsr a
009C9E  1  05 47           ora pdBlockNumberHigh
009CA0  1  85 47           sta pdBlockNumberHigh
009CA2  1  38              sec
009CA3  1  A5 10           lda BitmapBlock
009CA5  1  E5 0F           sbc BitmapBase
009CA7  1  0A              asl a
009CA8  1  0A              asl a
009CA9  1  0A              asl a
009CAA  1  0A              asl a
009CAB  1  05 47           ora pdBlockNumberHigh
009CAD  1  85 47           sta pdBlockNumberHigh
009CAF  1
009CAF  1  98              tya
009CB0  1  0A              asl a
009CB1  1  0A              asl a
009CB2  1  0A              asl a
009CB3  1  85 46           sta pdBlockNumberLow
009CB5  1  8A              txa
009CB6  1  05 46           ora pdBlockNumberLow
009CB8  1  85 46           sta pdBlockNumberLow
009CBA  1  AA              tax
009CBB  1  A5 47           lda pdBlockNumberHigh
009CBD  1  18              clc
009CBE  1  60              rts
009CBF  1
009CBF  1               BitmapClearMasks:
009CBF  1  7F              .byte %01111111
009CC0  1  BF              .byte %10111111
009CC1  1  DF              .byte %11011111
009CC2  1  EF              .byte %11101111
009CC3  1  F7              .byte %11110111
009CC4  1  FB              .byte %11111011
009CC5  1  FD              .byte %11111101
009CC6  1  FE              .byte %11111110
009CC7  1
009CC7  1               BitmapMasks:
009CC7  1  80              .byte %10000000
009CC8  1  40              .byte %01000000
009CC9  1  20              .byte %00100000
009CCA  1  10              .byte %00010000
009CCB  1  08              .byte %00001000
009CCC  1  04              .byte %00000100
009CCD  1  02              .byte %00000010
009CCE  1  01              .byte %00000001
009CCF  1
009CCF  1               ;--------------------------------------------------------------------------
009CCF  1               ; FreeOneBlockAX
009CCF  1               ;
009CCF  1               ; Input:  AX = block number to mark free
009CCF  1               ;
009CCF  1               ; Result: CLC for success (BitmapDirty is set, BitmapBuffer is updated)
009CCF  1               ;         SEC, A = error (eBakedBitmap if the block is invalid or already free)
009CCF  1               ;
009CCF  1               ; Destroys:  num
009CCF  1               ;
009CCF  1               ; You must have previously called ReadFirstBitmapBlock.
009CCF  1               ; Later, you have to call FlushBitmap.
009CCF  1               ;
009CCF  1               ; The bits in a block number locate a bit in the bitmap like this.  b = block
009CCF  1               ; offset of the bitmap block, H = 2nd half of the block, Y = byte offset,
009CCF  1               ; X = bit position:
009CCF  1               ;
009CCF  1               ;     b  b  b  b    H  y  y  y  y  y  y  y  y    x  x  x
009CCF  1               ;    15 14 13 12   11 10  9  8  7  6  5  4  3    2  1  0
009CCF  1               ;
009CCF  1               FreeOneBlockAX:
009CCF  1  85 1F           sta num+1
009CD1  1  86 1E           stx num
009CD3  1  4A              lsr a
009CD4  1  4A              lsr a
009CD5  1  4A              lsr a
009CD6  1  4A              lsr a
009CD7  1  20 2D 9C        jsr ReadBitmapBlockOffsetA     ; block offset = block/256 / 16
009CDA  1  B0 3D           bcs @exit
009CDC  1
009CDC  1  A5 1E           lda num
009CDE  1  29 07           and #7                         ; X = bits 0 to 2 of block number
009CE0  1  AA              tax
009CE1  1
009CE1  1  46 1F           lsr num+1
009CE3  1  66 1E           ror num
009CE5  1  46 1F           lsr num+1
009CE7  1  66 1E           ror num
009CE9  1  46 1F           lsr num+1
009CEB  1  66 1E           ror num
009CED  1  A4 1E           ldy num                        ; Y = bits 3 to 10 of block number
009CEF  1
009CEF  1  46 1F           lsr num+1                      ; bit 11 of block number puts us in 2nd half of a bitmap block
009CF1  1  B0 10           bcs @highHalf
009CF3  1
```

```
009CF3  1  B9 00 8E        lda BitmapBuffer,y
009CF6  1  1D C7 9C        ora BitmapMasks,x
009CF9  1  D9 00 8E        cmp BitmapBuffer,y
009CFC  1  F0 18           beq @bakedBitmap              ; block was already free
009CFE  1  99 00 8E        sta BitmapBuffer,y
009D01  1  D0 0E           bne @common
009D03  1
009D03  1               @highHalf:
009D03  1  B9 00 8F        lda BitmapBuffer+256,y
009D06  1  1D C7 9C        ora BitmapMasks,x
009D09  1  D9 00 8F        cmp BitmapBuffer+256,y
009D0C  1  F0 08           beq @bakedBitmap
009D0E  1  99 00 8F        sta BitmapBuffer+256,y
009D11  1
009D11  1               @common:
009D11  1  38              sec
009D12  1  66 11           ror BitmapDirty
009D14  1  18              clc
009D15  1  60              rts
009D16  1
009D16  1               @bakedBitmap:
009D16  1  A9 5A           lda #eBakedBitmap
009D18  1  38              sec
009D19  1               @exit:
009D19  1  60              rts
009D1A  1
009D1A  1               ;------------------------------------------------------------------------------
009D1A  1               ; FlushBitmap
009D1A  1               ;
009D1A  1               ; Writes the current bitmap block to disk, if it's dirty.
009D1A  1               ;
009D1A  1               FlushBitmap:
009D1A  1  24 11           bit BitmapDirty
009D1C  1  10 0C           bpl @clean
009D1E  1  46 11           lsr BitmapDirty
009D20  1  A9 00           lda #0
009D22  1  A6 10           ldx BitmapBlock
009D24  1  20 21 A7        jsr UseBitmapBuffer
009D27  1  4C 5F A7        jmp WriteBlockAX
009D2A  1               @clean:
009D2A  1  18              clc
009D2B  1  60              rts
009D2C  1
009D2C  1               ;------------------------------------------------------------------------------
009D2C  1               ; GetFileInfo
009D2C  1               ;
009D2C  1               ; Input:  Filename = pointer to name, starting with length byte ($05, "HELLO")
009D2C  1               ;
009D2C  1               ; Output: Filetype, Auxtype, FileSize (just 2 bytes)
009D2C  1               ;
009D2C  1               GetFileInfo:
009D2C  1  20 B7 9B        jsr UsePrefixNormally
009D2F  1  20 F3 A5        jsr LocateDirEntryForFilename
009D32  1  B0 1D           bcs @exit
009D34  1
009D34  1  A0 10           ldy #oFiletype
009D36  1  B1 0B           lda (EntryPtr),y
009D38  1  85 06           sta Filetype
009D3A  1
009D3A  1  A0 1F           ldy #oAuxtype
009D3C  1  B1 0B           lda (EntryPtr),y
009D3E  1  85 07           sta Auxtype
009D40  1  C8              iny
009D41  1  B1 0B           lda (EntryPtr),y
009D43  1  85 08           sta Auxtype+1
009D45  1
009D45  1  A0 15           ldy #oFileSize
009D47  1  B1 0B           lda (EntryPtr),y
009D49  1  85 09           sta FileSize
009D4B  1  C8              iny
009D4C  1  B1 0B           lda (EntryPtr),y
009D4E  1  85 0A           sta FileSize+1
009D50  1  18              clc
009D51  1               @exit:
009D51  1  60              rts
009D52  1
009D52  1               ;------------------------------------------------------------------------------
009D52  1               ; ReadFileAtAX
009D52  1               ;
009D52  1               ; Input:  Filename = pointer to name, starting with length byte ($05, "HELLO")
009D52  1               ;         AX = starting address ($0000 to use the file's Auxtype)
009D52  1               ;
009D52  1               ; ReadFileWithSpecialFirstBlock -- If the first block is special, we load it
009D52  1               ; as requested, and the other blocks are loaded as usual (starting at Destination).
009D52  1               ;
009D52  1               ReadFileAtAX:
009D52  1  85 01           sta Destination+1
009D54  1  86 00           stx Destination
009D56  1
009D56  1               API_ReadFile:
009D56  1               ReadFile:
009D56  1  A9 00           lda #0
009D58  1               ReadFileWithSpecialFirstBlock:
009D58  1  8D 02 87        sta SpecialFirstBlock
009D5B  1
009D5B  1  20 B7 9B        jsr UsePrefixNormally
009D5E  1  20 F3 A5        jsr LocateDirEntryForFilename
009D61  1  B0 48           bcs @exit
009D63  1
```

```
009D63  1  A0 00           ldy   #0
009D65  1  B1 0B           lda   (EntryPtr),y         ; storage type + length
009D67  1  C9 30           cmp   #kTree               ; only storage types 1 and 2 are OK
009D69  1  B0 4F           bcs   @badStorageType
009D6B  1  C9 11           cmp   #kSeedling+1
009D6D  1  90 4B           bcc   @badStorageType
009D6F  1
009D6F  1  A0 1E           ldy   #oAccess             ; make sure we're permitted Read access to this file
009D71  1  B1 0B           lda   (EntryPtr),y
009D73  1  29 01           and   #kAccessRead
009D75  1  D0 04           bne   @allowed
009D77  1  A9 4E           lda   #eFileLocked
009D79  1  38              sec
009D7A  1  60              rts
009D7B  1               @allowed:
009D7B  1
009D7B  1  A0 11           ldy   #oKeyBlock
009D7D  1  B1 0B           lda   (EntryPtr),y
009D7F  1  85 46           sta   pdBlockNumberLow
009D81  1  C8              iny
009D82  1  B1 0B           lda   (EntryPtr),y
009D84  1  85 47           sta   pdBlockNumberHigh
009D86  1
009D86  1  A0 15           ldy   #oFileSize
009D88  1  B1 0B           lda   (EntryPtr),y
009D8A  1  85 09           sta   FileSize
009D8C  1  C8              iny
009D8D  1  B1 0B           lda   (EntryPtr),y
009D8F  1  85 0A           sta   FileSize+1
009D91  1
009D91  1  A5 01           lda   Destination+1
009D93  1  05 00           ora   Destination
009D95  1  D0 0B           bne   @validate
009D97  1  A0 1F           ldy   #oAuxtype            ; use Auxtype for destination
009D99  1  B1 0B           lda   (EntryPtr),y
009D9B  1  85 00           sta   Destination
009D9D  1  C8              iny
009D9E  1  B1 0B           lda   (EntryPtr),y
009DA0  1  85 01           sta   Destination+1
009DA2  1               @validate:
009DA2  1  A5 01           lda   Destination+1
009DA4  1  C9 02           cmp   #2                   ; prohibit a read below address $0200
009DA6  1  B0 04           bcs   @destOK
009DA8  1  A9 56           lda   #eBadBufferAddr
009DAA  1  38              sec
009DAB  1               @exit:
009DAB  1  60              rts
009DAC  1               @destOK:
009DAC  1
009DAC  1  A0 00           ldy   #0
009DAE  1  B1 0B           lda   (EntryPtr),y         ;storage type + length
009DB0  1  29 F0           and   #kStorageTypeMask
009DB2  1  C9 10           cmp   #kSeedling
009DB4  1  F0 08           beq   @readSeedling
009DB6  1  C9 20           cmp   #kSapling
009DB8  1  F0 0A           beq   @readSapling
009DBA  1               @badStorageType:
009DBA  1  A9 4B           lda   #eBadStrgType
009DBC  1  38              sec
009DBD  1  60              rts
009DBE  1
009DBE  1               @readSeedling:
009DBE  1  20 36 A7        jsr   UseDestination
009DC1  1  4C 94 A7        jmp   DoReadBlock
009DC4  1
009DC4  1               @readSapling:
009DC4  1  20 19 A7        jsr   UseBuffer
009DC7  1  20 94 A7        jsr   DoReadBlock
009DCA  1  B0 DF           bcs   @exit
009DCC  1  20 36 A7        jsr   UseDestination
009DCF  1  A9 00           lda   #0
009DD1  1  85 1A           sta   BlockIndex
009DD3  1  AD 02 87        lda   SpecialFirstBlock
009DD6  1  D0 2A           bne   @specialFirstBlock
009DD8  1               @nextBlock:
009DD8  1  A4 1A           ldy   BlockIndex
009DDA  1  B9 00 8A        lda   buffer,y
009DDD  1  AA              tax
009DDE  1  B9 00 8B        lda   buffer+256,y
009DE1  1  20 3F A7        jsr   ReadBlockAXOrZeroes
009DE4  1  B0 C5           bcs   @exit
009DE6  1  E6 45           inc   pdIOBufferHigh
009DE8  1  E6 45           inc   pdIOBufferHigh
009DEA  1               @did1block:
009DEA  1  A5 0A           lda   FileSize+1
009DEC  1  C6 0A           dec   FileSize+1
009DEE  1  C6 0A           dec   FileSize+1
009DF0  1  C9 02           cmp   #2
009DF2  1  90 0A           bcc   @finished
009DF4  1  D0 04           bne   @moreWork
009DF6  1  A5 09           lda   FileSize
009DF8  1  F0 04           beq   @finished
009DFA  1               @moreWork:
009DFA  1  E6 1A           inc   BlockIndex
009DFC  1  D0 DA           bne   @nextBlock
009DFE  1
009DFE  1               @finished:
009DFE  1  A9 00           lda   #0
009E00  1  18              clc
```

```
009E01  1  60              rts
009E02  1
009E02  1                @specialFirstBlock:
009E02  1  20 2A A7         jsr UseDestinationOrSpecialBuffer
009E05  1  AD 00 8B         lda buffer+256
009E08  1  AE 00 8A         ldx buffer
009E0B  1  20 90 A7         jsr ReadBlockAX
009E0E  1  20 36 A7         jsr UseDestination
009E11  1  90 D7            bcc @did1block
009E13  1  60               rts
009E14  1
009E14  1                ;--------------------------------------------------------------------------
009E14  1                ; 4K boundary -- Move this up or down in the source code as needed, if the space
009E14  1                ;                before or after becomes full.
009E14  1                ;
009E14  1                ; The firmware lives at $9000, but during development it used to live at $A000.
009E14  1                ; Prevent anything bad from happening if someone accidentally calls one of the
009E14  1                ; old $A00x entry points.
009E14  1                ;--------------------------------------------------------------------------
009E14  1  xx xx xx xx      .res Origin+4096-*
009E18  1  xx xx xx xx

…

00A000  1  EA               nop
00A001  1  EA               nop
00A002  1  EA               nop
00A003  1  EA               nop
00A004  1  EA               nop
00A005  1  EA               nop
00A006  1  EA               nop
00A007  1  EA               nop
00A008  1  EA               nop
00A009  1  EA               nop
00A00A  1  EA               nop
00A00B  1  EA               nop
00A00C  1                 .if APPLE2
00A00C  1                   jmp     $FF69       ; Apple II monitor
00A00C  1                 .else
00A00C  1  4C 1F FF         jmp     APPLE1_MON
00A00F  1                 .endif
00A00F  1                ;--------------------------------------------------------------------------
00A00F  1
00A00F  1                ;--------------------------------------------------------------------------
00A00F  1                ; Rename
00A00F  1                ;
00A00F  1                ; Input:  OldFilename = original name
00A00F  1                ;         Filename = new name
00A00F  1                ;
00A00F  1                ; Result: CLC for success
00A00F  1                ;         SEC, A = error
00A00F  1                ;
00A00F  1                API_Rename:
00A00F  1                Rename:
00A00F  1  20 B7 9B         jsr UsePrefixNormally
00A012  1                ; first, find the original (file-not-found takes precedence over duplicate-file)
00A012  1  A6 05            ldx OldFilename+1
00A014  1  A4 04            ldy OldFilename
00A016  1  20 DD A5         jsr LocateDirEntryForNameXY
00A019  1  B0 52            bcs @exit
00A01B  1
00A01B  1                ; see if we're allowed to rename this file
00A01B  1  A0 1E            ldy #oAccess
00A01D  1  B1 0B            lda (EntryPtr),y
00A01F  1  29 40            and #kAccessRename
00A021  1  F0 4B            beq rename_no_access
00A023  1
00A023  1                ; make sure it's a storage type we can deal with (seedling, sapling, tree, directory)
00A023  1  A0 00            ldy #0
00A025  1  B1 0B            lda (EntryPtr),y
00A027  1  29 F0            and #kStorageTypeMask
00A029  1  85 06            sta Filetype              ; really storage type -- needed below for directories
00A02B  1  C9 31            cmp #kTree+1
00A02D  1  90 08            bcc @storage_type_ok
00A02F  1  C9 D0            cmp #kDirectory
00A031  1  F0 04            beq @storage_type_ok
00A033  1  A9 4B            lda #eBadStrgType
00A035  1  38               sec
00A036  1  60               rts
00A037  1                @storage_type_ok:
00A037  1
00A037  1                ; make sure there is no file already using the new name
00A037  1  20 F3 A5         jsr LocateDirEntryForFilename
00A03A  1  90 2E            bcc @duplicate            ; oops, it already exists
00A03C  1  C9 46            cmp #eFileNotFound        ; we expect file-not-found, and any other error means we're done
00A03E  1  D0 2C            bne @error
00A040  1
00A040  1  A6 05            ldx OldFilename+1
00A042  1  A4 04            ldy OldFilename
00A044  1  20 DD A5         jsr LocateDirEntryForNameXY
00A047  1  B0 24            bcs @exit
00A049  1
00A049  1                ; replace the entry's filename, updating the length (but leaving the storage type untouched)
00A049  1  A0 00            ldy #0
00A04B  1  B1 02            lda (Filename),y        ; length
00A04D  1  A8               tay
00A04E  1                @copy:
00A04E  1  B1 02            lda (Filename),y
00A050  1  91 0B            sta (EntryPtr),y
```

```
00A052  1  88             dey
00A053  1  D0 F9           bne @copy
00A055  1
00A055  1  B1 0B           lda (EntryPtr),y
00A057  1  29 F0           and #kStorageTypeMask
00A059  1  11 02           ora (Filename),y        ; update the name length, leaving storage type unchanged
00A05B  1  91 0B           sta (EntryPtr),y
00A05D  1  20 58 A7        jsr WriteCurrentDirectoryBlock
00A060  1  B0 0B           bcs @exit
00A062  1
00A062  1  A6 06           ldx Filetype           ; storage type
00A064  1  E0 D0           cpx #kDirectory
00A066  1  F0 0A           beq finishRenamingDirectory
00A068  1  18              clc
00A069  1                ;  lda #0                 ; A is still zero from WriteBlock
00A069  1  60              rts
00A06A  1
00A06A  1                @duplicate:
00A06A  1  A9 47           lda #eDuplicateFile
00A06C  1                @error:
00A06C  1  38              sec
00A06D  1                @exit:
00A06D  1  60              rts
00A06E  1                rename_no_access:
00A06E  1                delete_no_access:
00A06E  1  A9 4E           lda #eFileLocked
00A070  1  38              sec
00A071  1  60              rts
00A072  1
00A072  1                finishRenamingDirectory:
00A072  1  20 19 A7        jsr UseBuffer
00A075  1  20 A7 9B        jsr LoadKeyBlockIntoAX
00A078  1  20 90 A7        jsr ReadBlockAX
00A07B  1  B0 17           bcs @exit
00A07D  1
00A07D  1                ; replace the filename in the directory header
00A07D  1  A0 00           ldy #0
00A07F  1  B1 02           lda (Filename),y        ; length
00A081  1  A8              tay
00A082  1                @copy:
00A082  1  B1 02           lda (Filename),y
00A084  1  99 04 8A        sta buffer+oVolStorageType,y
00A087  1  88              dey
00A088  1  D0 F8           bne @copy
00A08A  1  B1 02           lda (Filename),y        ;length
00A08C  1  09 E0           ora #kSubdirHeader
00A08E  1  8D 04 8A        sta buffer+oVolStorageType
00A091  1  4C 63 A7        jmp DoWriteBlock
00A094  1                @exit:
00A094  1  60              rts
00A095  1
00A095  1
00A095  1                ;--------------------------------------------------------------------------------
00A095  1                ; Delete
00A095  1                ;
00A095  1                ; Input:  Filename
00A095  1                ; Result: CLC for success
00A095  1                ;         SEC, A = error
00A095  1                ;
00A095  1                API_Delete:
00A095  1                Delete:
00A095  1  20 B7 9B        jsr UsePrefixNormally
00A098  1  20 F3 A5        jsr LocateDirEntryForFilename
00A09B  1  90 01           bcc @continue
00A09D  1                @exit:
00A09D  1  60              rts
00A09E  1                @continue:
00A09E  1
00A09E  1                ; see if we're permitted to delete this file
00A09E  1  A0 1E           ldy #oAccess
00A0A0  1  B1 0B           lda (EntryPtr),y
00A0A2  1  29 80           and #kAccessDelete
00A0A4  1  F0 C8           beq delete_no_access
00A0A6  1
00A0A6  1  20 27 9C        jsr ReadFirstBitmapBlock
00A0A9  1  B0 F2           bcs @exit
00A0AB  1
00A0AB  1                ; see if it's a storage type we know how to delete (seedling or sapling)
00A0AB  1  A0 00           ldy #0
00A0AD  1  B1 0B           lda (EntryPtr),y
00A0AF  1  29 F0           and #kStorageTypeMask
00A0B1  1  C9 10           cmp #kSeedling
00A0B3  1  F0 3D           beq @delete_seedling
00A0B5  1  C9 20           cmp #kSapling
00A0B7  1  F0 43           beq @delete_sapling
00A0B9  1  C9 D0           cmp #kDirectory
00A0BB  1  F0 04           beq @delete_directory
00A0BD  1  A9 4B           lda #eBadStrgType
00A0BF  1  38              sec
00A0C0  1                @exit2:
00A0C0  1  60              rts
00A0C1  1
00A0C1  1                @delete_directory:
00A0C1  1  20 19 A7        jsr UseBuffer
00A0C4  1  20 A7 9B        jsr LoadKeyBlockIntoAX
00A0C7  1  20 90 A7        jsr ReadBlockAX
00A0CA  1  B0 F4           bcs @exit2
00A0CC  1                ; we can only delete the directory if it's empty
00A0CC  1  AD 25 8A        lda buffer+oVolFileCount
```

```
00A0CF  1  0D 26 8A        ora buffer+oVolFileCount+1
00A0D2  1  D0 9A           bne delete_no_access
00A0D4  1                @freeNextDirBlock:
00A0D4  1  A5 47           lda pdBlockNumberHigh
00A0D6  1  A6 46           ldx pdBlockNumberLow
00A0D8  1  20 CF 9C        jsr FreeOneBlockAX
00A0DB  1  B0 E3           bcs @exit2
00A0DD  1  AD 03 8A        lda buffer+oDirLinkNext+1
00A0E0  1  0D 02 8A        ora buffer+oDirLinkNext
00A0E3  1  F0 47           beq @common
00A0E5  1  AD 03 8A        lda buffer+oDirLinkNext+1
00A0E8  1  AE 02 8A        ldx buffer+oDirLinkNext
00A0EB  1  20 90 A7        jsr ReadBlockAX
00A0EE  1  90 E4           bcc @freeNextDirBlock
00A0F0  1  B0 CE           bcs @exit2
00A0F2  1
00A0F2  1                @delete_seedling:
00A0F2  1  20 A7 9B        jsr LoadKeyBlockIntoAX
00A0F5  1  20 CF 9C        jsr FreeOneBlockAX
00A0F8  1                @exit3:
00A0F8  1  B0 C6           bcs @exit2
00A0FA  1  90 30           bcc @common
00A0FC  1
00A0FC  1                @delete_sapling:
00A0FC  1                ; read the index block
00A0FC  1  20 A7 9B        jsr LoadKeyBlockIntoAX
00A0FF  1  20 19 A7        jsr UseBuffer
00A102  1  20 90 A7        jsr ReadBlockAX
00A105  1  B0 B9           bcs @exit2
00A107  1                ; free the data blocks
00A107  1  A0 00           ldy #0
00A109  1                @loop:
00A109  1  B9 00 8A        lda buffer,y
00A10C  1  19 00 8B        ora buffer+256,y
00A10F  1  F0 10           beq @skip
00A111  1  84 1A           sty BlockIndex
00A113  1  B9 00 8A        lda buffer,y
00A116  1  AA              tax
00A117  1  B9 00 8B        lda buffer+256,y
00A11A  1  20 CF 9C        jsr FreeOneBlockAX
00A11D  1  B0 A1           bcs @exit2
00A11F  1  A4 1A           ldy BlockIndex
00A121  1                @skip:
00A121  1  C8              iny
00A122  1  D0 E5           bne @loop
00A124  1                ; free the index block
00A124  1  20 A7 9B        jsr LoadKeyBlockIntoAX
00A127  1  20 CF 9C        jsr FreeOneBlockAX
00A12A  1  B0 CC           bcs @exit3
00A12C  1
00A12C  1                @common:
00A12C  1                ; Change the directory entry's storage type and name length to 0
00A12C  1                ; (leaving the length in place makes Mr.Fixit claim "incomplete delete")
00A12C  1  A0 00           ldy #0
00A12E  1  98              tya
00A12F  1  91 0B           sta (EntryPtr),y
00A131  1
00A131  1                FlushBitmapAndDecrementFileCount:
00A131  1  20 1A 9D        jsr FlushBitmap
00A134  1  B0 14           bcs @exit
00A136  1  20 69 A7        jsr FlushAndReadFirstDirectoryBlock
00A139  1
00A139  1                ; decrement the file count in the directory header
00A139  1  38              sec
00A13A  1  AD 25 8C        lda DirectoryBuffer+oVolFileCount
00A13D  1  E9 01           sbc #1
00A13F  1  8D 25 8C        sta DirectoryBuffer+oVolFileCount
00A142  1  B0 03           bcs @noBorrow
00A144  1  CE 26 8C        dec DirectoryBuffer+oVolFileCount+1
00A147  1                @noBorrow:
00A147  1  4C 58 A7        jmp WriteCurrentDirectoryBlock
00A14A  1
00A14A  1                @exit:
00A14A  1  60              rts
00A14B  1
00A14B  1                ;================================================================================
00A14B  1                ;--------------------------------------------------------------------------------
00A14B  1                ; $A240 = CALL -24000  -- convenient entry point from BASIC.
00A14B  1                ;
00A14B  1                ; Move this up or down in the source code as needed, if the space before or
00A14B  1                ; after becomes full.
00A14B  1                ;--------------------------------------------------------------------------------
00A14B  1                 .if APPLE2
00A14B  1                 .else
00A14B  1  xx xx xx xx      .res $A240-*
00A14F  1  xx xx xx xx

…

00A23F  1  xx
00A240  1  4C DC 90        jmp MenuExitToBASIC
00A243  1                 .endif
00A243  1                ;--------------------------------------------------------------------------------
00A243  1                ;================================================================================
00A243  1
00A243  1
00A243  1
00A243  1                ;--------------------------------------------------------------------------------
00A243  1                ; WriteFile
```

```
00A243  1              ;
00A243  1              ; Input:  Filename
00A243  1              ;         Destination = starting address
00A243  1              ;         FileSize    = number of bytes to write
00A243  1              ;         Filetype
00A243  1              ;         Auxtype
00A243  1              ; Result: CLC for success
00A243  1              ;         SEC, A = error
00A243  1              ;
00A243  1              ; Creates a new file, or replaces an existing file with the same name.
00A243  1              ;
00A243  1              API_WriteFile:
00A243  1              WriteFile:
00A243  1  A9 00          lda #0
00A245  1              WriteFileWithSpecialFirstBlock:
00A245  1  8D 02 87       sta SpecialFirstBlock
00A248  1  20 95 A0       jsr Delete           ; note that this validates the Filename syntax & uppercases it
00A24B  1  90 06          bcc @continue
00A24D  1  C9 46          cmp #eFileNotFound   ; file-not-found is OK, but any other error means we're done
00A24F  1  F0 02          beq @continue
00A251  1  38             sec
00A252  1              @exit:
00A252  1  60             rts
00A253  1              @continue:
00A253  1
00A253  1              ; See if the disk has enough free blocks for our file
00A253  1  20 BD 9B       jsr ComputeFreeBlocksUpTo256
00A256  1  B0 FA          bcs @exit
00A258  1  20 03 A7       jsr BlocksNeededFromFileSize
00A25B  1  85 15          sta UsedBlocks        ; now "blocks needed" (1 byte)
00A25D  1
00A25D  1  A5 18          lda FreeBlocks+1
00A25F  1  D0 0A          bne @enough_room      ; 256 or more free blocks is plenty
00A261  1  A5 17          lda FreeBlocks
00A263  1  C5 15          cmp UsedBlocks
00A265  1  B0 04          bcs @enough_room
00A267  1  A9 48          lda #eVolumeFull
00A269  1  38             sec
00A26A  1  60             rts
00A26B  1              @enough_room:
00A26B  1
00A26B  1  20 18 A6       jsr LocateAvailableDirEntry
00A26E  1  B0 E2          bcs @exit
00A270  1
00A270  1  A0 26          ldy #kDirEntrySize-1
00A272  1  A9 00          lda #0
00A274  1              @zero:
00A274  1  91 0B          sta (EntryPtr),y
00A276  1  88             dey
00A277  1  10 FB          bpl @zero
00A279  1
00A279  1  A0 00          ldy #0
00A27B  1  B1 02          lda (Filename),y
00A27D  1  A8             tay
00A27E  1              @copyName:
00A27E  1  B1 02          lda (Filename),y
00A280  1  91 0B          sta (EntryPtr),y
00A282  1  88             dey
00A283  1  10 F9          bpl @copyName
00A285  1
00A285  1              ; set the filetype & auxtype
00A285  1  A0 10          ldy #oFiletype
00A287  1  A5 06          lda Filetype
00A289  1  91 0B          sta (EntryPtr),y
00A28B  1
00A28B  1  A0 1F          ldy #oAuxtype
00A28D  1  A5 07          lda Auxtype
00A28F  1  91 0B          sta (EntryPtr),y
00A291  1  C8             iny
00A292  1  A5 08          lda Auxtype+1
00A294  1  91 0B          sta (EntryPtr),y
00A296  1
00A296  1              ; set the block count
00A296  1  A0 13          ldy #oBlockCount
00A298  1  A5 15          lda UsedBlocks
00A29A  1  91 0B          sta (EntryPtr),y
00A29C  1  C8             iny
00A29D  1  A9 00          lda #0
00A29F  1  91 0B          sta (EntryPtr),y
00A2A1  1
00A2A1  1              ; set the storage type to Seeding or Sapling (use UsedBlocks for this)
00A2A1  1  A9 10          lda #kSeedling
00A2A3  1  A6 15          ldx UsedBlocks
00A2A5  1  E0 02          cpx #2
00A2A7  1  90 02          bcc @setStorageType
00A2A9  1  A9 20          lda #kSapling
00A2AB  1              @setStorageType:
00A2AB  1  A0 00          ldy #0
00A2AD  1  11 0B          ora (EntryPtr),y
00A2AF  1  91 0B          sta (EntryPtr),y
00A2B1  1
00A2B1  1              ; set the access to Unlocked
00A2B1  1  A0 1E          ldy #oAccess
00A2B3  1  A9 E3          lda #kAccessFull+kAccessNeedsBackup
00A2B5  1  91 0B          sta (EntryPtr),y
00A2B7  1
00A2B7  1              ; set the end of file (3 bytes) from FileSize
00A2B7  1  A0 15          ldy #oFileSize
00A2B9  1  A5 09          lda FileSize
```

```
00A2BB  1  91 0B           sta (EntryPtr),y
00A2BD  1  C8              iny
00A2BE  1  A5 0A           lda FileSize+1
00A2C0  1  91 0B           sta (EntryPtr),y
00A2C2  1
00A2C2  1               ; set Version, Minimum version
00A2C2  1  A0 1C           ldy #oVersion
00A2C4  1  A9 00           lda #0
00A2C6  1  91 0B           sta (EntryPtr),y    ; version = 0
00A2C8  1  C8              iny
00A2C9  1  91 0B           sta (EntryPtr),y    ; min version = 0
00A2CB  1
00A2CB  1               ; set the Header Pointer field to the first block of the directory
00A2CB  1  A0 25           ldy #oHeaderPointer
00A2CD  1  AD AD 87        lda FirstDirectoryBlock
00A2D0  1  91 0B           sta (EntryPtr),y
00A2D2  1  C8              iny
00A2D3  1  AD AE 87        lda FirstDirectoryBlock+1
00A2D6  1  91 0B           sta (EntryPtr),y
00A2D8  1
00A2D8  1               ; allocate the key block & store it into the directory entry
00A2D8  1  20 27 9C        jsr ReadFirstBitmapBlock
00A2DB  1  B0 1B           bcs @exit2
00A2DD  1  20 5D 9C        jsr AllocateOneBlockAX    ;also sets pdBlockNumber
00A2E0  1  B0 16           bcs @exit2
00A2E2  1
00A2E2  1  A0 12           ldy #oKeyBlock+1
00A2E4  1  91 0B           sta (EntryPtr),y
00A2E6  1  8A              txa
00A2E7  1  88              dey
00A2E8  1  91 0B           sta (EntryPtr),y
00A2EA  1
00A2EA  1  A5 15           lda UsedBlocks
00A2EC  1  C9 02           cmp #2
00A2EE  1  B0 09           bcs @writeSapling
00A2F0  1
00A2F0  1               @writeSeedling:
00A2F0  1  20 36 A7        jsr UseDestination
00A2F3  1  20 63 A7        jsr DoWriteBlock
00A2F6  1  90 3D           bcc @updateFileCount
00A2F8  1               @exit2:
00A2F8  1  60              rts
00A2F9  1
00A2F9  1               @writeSapling:
00A2F9  1  20 D0 A5        jsr ZeroBuffer
00A2FC  1  85 1A           sta BlockIndex            ;zero
00A2FE  1
00A2FE  1               @loop:
00A2FE  1  20 5D 9C        jsr AllocateOneBlockAX    ;also sets pdBlockNumber
00A301  1  B0 F5           bcs @exit2
00A303  1
00A303  1  A4 1A           ldy BlockIndex
00A305  1  99 00 8B        sta buffer+256,y
00A308  1  8A              txa
00A309  1  99 00 8A        sta buffer,y
00A30C  1
00A30C  1  20 2A A7        jsr UseDestinationOrSpecialBuffer
00A30F  1  20 63 A7        jsr DoWriteBlock
00A312  1  B0 E4           bcs @exit2
00A314  1
00A314  1  E6 01           inc Destination+1
00A316  1  E6 01           inc Destination+1
00A318  1  E6 1A           inc BlockIndex
00A31A  1  A5 0A           lda FileSize+1
00A31C  1  C6 0A           dec FileSize+1
00A31E  1  C6 0A           dec FileSize+1
00A320  1  C9 02           cmp #2
00A322  1  90 06           bcc @finished
00A324  1  D0 D8           bne @loop
00A326  1  A5 09           lda FileSize
00A328  1  D0 D4           bne @loop
00A32A  1               @finished:
00A32A  1
00A32A  1               ; write index block (its block number is in the directory entry at oKeyBlock)
00A32A  1  20 19 A7        jsr UseBuffer
00A32D  1  20 A7 9B        jsr LoadKeyBlockIntoAX
00A330  1  20 5F A7        jsr WriteBlockAX
00A333  1  B0 C3           bcs @exit2
00A335  1
00A335  1               @updateFileCount:
00A335  1               FlushBitmapAndIncrementFileCount:
00A335  1  20 1A 9D        jsr FlushBitmap
00A338  1  B0 0E           bcs @exit
00A33A  1  20 69 A7        jsr FlushAndReadFirstDirectoryBlock
00A33D  1
00A33D  1               ; increment the file count in the directory header
00A33D  1  EE 25 8C        inc DirectoryBuffer+oVolFileCount
00A340  1  D0 03           bne @noCarry
00A342  1  EE 26 8C        inc DirectoryBuffer+oVolFileCount+1
00A345  1               @noCarry:
00A345  1  4C 58 A7        jmp WriteCurrentDirectoryBlock
00A348  1
00A348  1               @exit:
00A348  1  60              rts
00A349  1
00A349  1               ;--------------------------------------------------------------------------------
00A349  1               ; LoadBASICFile
00A349  1               ;
00A349  1               ; Input:  Filename
```

```
00A349  1               ;
00A349  1               ; We need to load the first block of the file into StagingBuffer, and then copy
00A349  1               ; part of it into zero page at $4A..FF.  The rest of the file loads starting
00A349  1               ; at the file's auxtype, which should match the stored LOMEM value.
00A349  1               ;
00A349  1               ; We call ReadFile with a special request that it read the first block into
00A349  1               ; StagingBuffer.
00A349  1               ;
00A349  1               API_LoadBASICFile:
00A349  1               LoadBASICFile:
00A349  1  20 B7 9B        jsr UsePrefixNormally
00A34C  1  A9 00           lda #0
00A34E  1  85 01           sta Destination+1
00A350  1  85 00           sta Destination
00A352  1  A9 88           lda #>StagingBuffer
00A354  1  20 58 9D        jsr ReadFileWithSpecialFirstBlock
00A357  1  B0 1E           bcs @exit
00A359  1
00A359  1  A0 10           ldy #oFiletype
00A35B  1  B1 0B           lda (EntryPtr),y
00A35D  1  C9 F1           cmp #kFiletypeBASIC1
00A35F  1  D0 13           bne @unknownFormat
00A361  1
00A361  1  AD 00 88        lda StagingBuffer
00A364  1  C9 41           cmp #'A'
00A366  1  D0 0C           bne @unknownFormat
00A368  1  AD 01 88        lda StagingBuffer+1
00A36B  1  C9 31           cmp #'1'
00A36D  1  D0 05           bne @unknownFormat
00A36F  1  AD 02 88        lda StagingBuffer+2   ; version, must be 0
00A372  1  F0 04           beq @ok
00A374  1               @unknownFormat:
00A374  1  A9 FE           lda #eUnknownBASICFormat
00A376  1  38              sec
00A377  1               @exit:
00A377  1  60              rts
00A378  1               @ok:
00A378  1
00A378  1  A2 4A           ldx #$4A
00A37A  1               @copyZP:
00A37A  1  BD 00 88        lda StagingBuffer,x
00A37D  1                .if APPLE2
00A37D  1                  sta $800,x
00A37D  1                .else
00A37D  1  95 00           sta 0,x
00A37F  1                .endif
00A37F  1  E8              inx
00A380  1  D0 F8           bne @copyZP
00A382  1
00A382  1  A9 00           lda #0
00A384  1  18              clc
00A385  1  60              rts
00A386  1
00A386  1               ;------------------------------------------------------------------------------
00A386  1               ; SaveBASICFile
00A386  1               ;
00A386  1               ; Input:  Filename
00A386  1               ;
00A386  1               ; We need to save zero page from $4A..FF, and all the memory between LOMEM
00A386  1               ; and HIMEM.  We'll call WriteFile with a special request that it write from
00A386  1               ; StagingBuffer as the first block.  The FileSize and Destination address we
00A386  1               ; request allow for the extra data in StagingBuffer.
00A386  1               ;
00A386  1               API_SaveBASICFile:
00A386  1               SaveBASICFile:
00A386  1  A9 00           lda #0
00A388  1  AA              tax
00A389  1               @zero:
00A389  1  9D 00 88        sta StagingBuffer,x
00A38C  1  9D 00 89        sta StagingBuffer+256,x
00A38F  1  CA              dex
00A390  1  D0 F7           bne @zero
00A392  1
00A392  1  A9 41           lda #'A'
00A394  1  8D 00 88        sta StagingBuffer
00A397  1  A9 31           lda #'1'
00A399  1  8D 01 88        sta StagingBuffer+1
00A39C  1               ; StagingBuffer+2 is the file format version.  Leave it as 0 for now.
00A39C  1
00A39C  1  A2 4A           ldx #$4A
00A39E  1               @copyZP:
00A39E  1  B5 00           lda 0,x
00A3A0  1  9D 00 88        sta StagingBuffer,x
00A3A3  1  E8              inx
00A3A4  1  D0 F8           bne @copyZP
00A3A6  1
00A3A6  1  A6 4B           ldx LOMEM+1
00A3A8  1  A4 4A           ldy LOMEM
00A3AA  1  86 08           stx Auxtype+1
00A3AC  1  84 07           sty Auxtype
00A3AE  1  CA              dex
00A3AF  1  CA              dex               ; subtract 2 to allow for extra buffer
00A3B0  1  86 01           stx Destination+1
00A3B2  1  84 00           sty Destination
00A3B4  1
00A3B4  1  38              sec
00A3B5  1  A5 4C           lda HIMEM
00A3B7  1  E5 4A           sbc LOMEM
00A3B9  1  85 09           sta FileSize
```

```
00A3BB  1  A5 4D           lda HIMEM+1
00A3BD  1  E5 4B           sbc LOMEM+1
00A3BF  1  90 0D           bcc @error
00A3C1  1  69 01           adc #1             ; adds 2, since carry is set
00A3C3  1  85 0A           sta FileSize+1
00A3C5  1
00A3C5  1  A9 F1           lda #kFiletypeBASIC1
00A3C7  1  85 06           sta Filetype
00A3C9  1  A9 88           lda #>StagingBuffer
00A3CB  1  4C 45 A2        jmp WriteFileWithSpecialFirstBlock
00A3CE  1
00A3CE  1               @error:
00A3CE  1  A9 56           lda #eBadBufferAddr
00A3D0  1  38              sec
00A3D1  1  60              rts
00A3D2  1
00A3D2  1
00A3D2  1               ;---------------------------------------------------------------------------
00A3D2  1               ; NewDirectoryAtRoot
00A3D2  1               ;
00A3D2  1               ; Input:   Filename = name of directory
00A3D2  1               ; Output:  CLC for success, SEC/A=error
00A3D2  1               ;
00A3D2  1               API_NewDirectoryAtRoot:
00A3D2  1               NewDirectoryAtRoot:
00A3D2  1  20 B0 9B        jsr DoNotUsePrefix
00A3D5  1  20 F3 A5        jsr LocateDirEntryForFilename
00A3D8  1  20 B7 9B        jsr UsePrefixNormally
00A3DB  1  90 06           bcc @dup
00A3DD  1  C9 46           cmp #eFileNotFound
00A3DF  1  F0 06           beq @continue
00A3E1  1  38              sec
00A3E2  1  60              rts
00A3E3  1               @dup:
00A3E3  1  A9 47           lda #eDuplicateFile
00A3E5  1  38              sec
00A3E6  1               @exit:
00A3E6  1  60              rts
00A3E7  1
00A3E7  1               @continue:
00A3E7  1  20 B0 9B        jsr DoNotUsePrefix
00A3EA  1  20 18 A6        jsr LocateAvailableDirEntry
00A3ED  1  20 B7 9B        jsr UsePrefixNormally
00A3F0  1  B0 F4           bcs @exit
00A3F2  1
00A3F2  1               ; zero out the new directory entry
00A3F2  1  A0 26           ldy #kDirEntrySize-1
00A3F4  1  A9 00           lda #0
00A3F6  1               @zero:
00A3F6  1  91 0B           sta (EntryPtr),y
00A3F8  1  88              dey
00A3F9  1  10 FB           bpl @zero
00A3FB  1
00A3FB  1               ; construct the dir entry (EntryPtr) in parallel with the directory header (Buffer)
00A3FB  1  20 D0 A5        jsr ZeroBuffer
00A3FE  1
00A3FE  1               ; allocate the new dir's key block & store it into the directory entry
00A3FE  1  20 27 9C        jsr ReadFirstBitmapBlock
00A401  1  B0 E3           bcs @exit
00A403  1  20 5D 9C        jsr AllocateOneBlockAX    ;also sets pdBlockNumber
00A406  1               @exit1:
00A406  1  B0 DE           bcs @exit
00A408  1
00A408  1  A0 12           ldy #oKeyBlock+1
00A40A  1  91 0B           sta (EntryPtr),y
00A40C  1  8A              txa
00A40D  1  88              dey
00A40E  1  91 0B           sta (EntryPtr),y
00A410  1
00A410  1               ; set the name
00A410  1  A0 00           ldy #0
00A412  1  B1 02           lda (Filename),y
00A414  1  A8              tay
00A415  1               @copyName:
00A415  1  B1 02           lda (Filename),y
00A417  1  91 0B           sta (EntryPtr),y
00A419  1  99 04 8A        sta buffer+oVolStorageType,y
00A41C  1  88              dey
00A41D  1  10 F6           bpl @copyName
00A41F  1
00A41F  1               ; set the storage type to Directory
00A41F  1  A9 D0           lda #kDirectory
00A421  1  A0 00           ldy #0
00A423  1  11 0B           ora (EntryPtr),y
00A425  1  91 0B           sta (EntryPtr),y
00A427  1               ; set the dir header's storage type
00A427  1  A9 E0           lda #kSubdirHeader
00A429  1  0D 04 8A        ora buffer+oVolStorageType
00A42C  1  8D 04 8A        sta buffer+oVolStorageType
00A42F  1
00A42F  1               ; set the filetype
00A42F  1  A0 10           ldy #oFiletype
00A431  1  A9 0F           lda #kFiletypeDirectory
00A433  1  91 0B           sta (EntryPtr),y
00A435  1
00A435  1               ; set the block count to 1
00A435  1  A0 13           ldy #oBlockCount
00A437  1  A9 01           lda #1
00A439  1  91 0B           sta (EntryPtr),y
```

```
00A43B  1
00A43B  1                ; set the access to Unlocked
00A43B  1  A0 1E            ldy #oAccess
00A43D  1  A9 E3            lda #kAccessFull+kAccessNeedsBackup
00A43F  1  91 0B            sta (EntryPtr),y
00A441  1  8D 22 8A         sta buffer+oVolAccess
00A444  1
00A444  1                ; set the end of file (3 bytes) to $0200
00A444  1  A0 16            ldy #oFileSize+1
00A446  1  A9 02            lda #$02
00A448  1  91 0B            sta (EntryPtr),y
00A44A  1
00A44A  1                ; set the Header Pointer field to the first block of the (root) directory
00A44A  1  A0 25            ldy #oHeaderPointer
00A44C  1  AD AD 87         lda FirstDirectoryBlock
00A44F  1  91 0B            sta (EntryPtr),y
00A451  1  C8               iny
00A452  1  AD AE 87         lda FirstDirectoryBlock+1
00A455  1  91 0B            sta (EntryPtr),y
00A457  1
00A457  1                ; set entry_length, parent_entry_length
00A457  1  A9 27            lda #kDirEntrySize
00A459  1  8D 23 8A         sta buffer+oVolEntryLength
00A45C  1  8D 2A 8A         sta buffer+oSubdirParentEntryLength
00A45F  1                ; set entries per block
00A45F  1  A9 0D            lda #kEntriesPerBlock
00A461  1  8D 24 8A         sta buffer+oVolEntriesPerBlock
00A464  1
00A464  1                ; parent_pointer (block #), parent_entry_number (subdir header knows where its directory entry is)
00A464  1  A5 0E            lda DirectoryBlock+1
00A466  1  A6 0D            ldx DirectoryBlock
00A468  1  8D 28 8A         sta buffer+oSubdirParentPointer+1
00A46B  1  8E 27 8A         stx buffer+oSubdirParentPointer
00A46E  1
00A46E  1                ; compute parent entry number = entries-per-block + 1 - EntryCounter
00A46E  1  38               sec
00A46F  1  A9 0E            lda #kEntriesPerBlock+1
00A471  1  E5 19            sbc EntryCounter
00A473  1  8D 29 8A         sta buffer+oSubdirParentEntryNum
00A476  1
00A476  1                ; finish up by writing the directory's first block and the directory entry
00A476  1  20 19 A7         jsr UseBuffer
00A479  1  20 63 A7         jsr DoWriteBlock
00A47C  1  B0 88            bcs @exit1
00A47E  1
00A47E  1  4C 35 A3         jmp FlushBitmapAndIncrementFileCount
00A481  1
00A481  1
00A481  1                ;--------------------------------------------------------------------------------
00A481  1                ; SetPrefix
00A481  1                ;
00A481  1                ; Input:   Filename = name of directory (empty = use root)
00A481  1                ;
00A481  1                SetPrefix:
00A481  1  A0 00            ldy #0
00A483  1  B1 02            lda (Filename),y
00A485  1  F0 10            beq @emptyPrefix
00A487  1
00A487  1  20 B0 9B         jsr DoNotUsePrefix
00A48A  1  20 F3 A5         jsr LocateDirEntryForFilename
00A48D  1  20 B7 9B         jsr UsePrefixNormally
00A490  1  B0 12            bcs @error
00A492  1  20 D9 9A         jsr RequireDirectory
00A495  1  B0 0D            bcs @error
00A497  1
00A497  1                @emptyPrefix:
00A497  1  A0 0F            ldy #15
00A499  1                @copyPrefix:
00A499  1  B1 02            lda (Filename),y
00A49B  1  99 B0 87         sta PrefixDirectory,y
00A49E  1  88               dey
00A49F  1  10 F8            bpl @copyPrefix
00A4A1  1
00A4A1  1  A9 00            lda #0
00A4A3  1  38               sec
00A4A4  1                @error:
00A4A4  1  60               rts
00A4A5  1
00A4A5  1
00A4A5  1                ;--------------------------------------------------------------------------------
00A4A5  1                ; FormatDrive
00A4A5  1                ;
00A4A5  1                ; Input:   Filename = volume name
00A4A5  1                ;          DriveNumber
00A4A5  1                ;
00A4A5  1                FormatDrive:
00A4A5  1  20 C7 A6         jsr SyntaxCheckFilename
00A4A8  1  B0 13            bcs @exit
00A4AA  1
00A4AA  1                ; GetStatus for the block count.
00A4AA  1  A9 00            lda #PRODOS_STATUS
00A4AC  1  85 42            sta pdCommandCode
00A4AE  1  20 98 A7         jsr CFBlockDriver
00A4B1  1  B0 0A            bcs @exit
00A4B3  1  84 14            sty TotalBlocks+1
00A4B5  1  86 13            stx TotalBlocks
00A4B7  1  98               tya
00A4B8  1  D0 04            bne @enoughBlocks
00A4BA  1  A9 2D            lda #PRODOS_BADBLOCK
```

```
00A4BC  1  38            sec
00A4BD  1              @exit:
00A4BD  1  60            rts
00A4BE  1              @enoughBlocks:
00A4BE  1
00A4BE  1              ; Zero out blocks 0 through 64
00A4BE  1  20 D0 A5      jsr ZeroBuffer
00A4C1  1  20 19 A7      jsr UseBuffer
00A4C4  1  A9 00         lda #0
00A4C6  1  A2 40         ldx #64
00A4C8  1  85 47         sta pdBlockNumberHigh
00A4CA  1  86 46         stx pdBlockNumberLow
00A4CC  1              @zeroBlocks:
00A4CC  1  20 63 A7      jsr DoWriteBlock
00A4CF  1  B0 EC         bcs @exit
00A4D1  1  C6 46         dec pdBlockNumberLow
00A4D3  1  10 F7         bpl @zeroBlocks
00A4D5  1
00A4D5  1              ; Construct and write block 2, the main directory block.
00A4D5  1  A0 00         ldy #0
00A4D7  1  B1 02         lda (Filename),y
00A4D9  1  A8            tay
00A4DA  1              @copyName:
00A4DA  1  B1 02         lda (Filename),y
00A4DC  1  99 04 8A      sta buffer+oVolStorageType,y
00A4DF  1  88            dey
00A4E0  1  10 F8         bpl @copyName
00A4E2  1  09 F0         ora #kVolume
00A4E4  1  8D 04 8A      sta buffer+oVolStorageType
00A4E7  1
00A4E7  1  A9 03         lda #kRootDirectoryBlock+1
00A4E9  1  8D 02 8A      sta buffer+oDirLinkNext
00A4EC  1  A9 05         lda #5
00A4EE  1  8D 20 8A      sta buffer+oVolVersion
00A4F1  1  A9 E3         lda #kAccessFull+kAccessNeedsBackup
00A4F3  1  8D 22 8A      sta buffer+oVolAccess
00A4F6  1  A9 27         lda #kDirEntrySize
00A4F8  1  8D 23 8A      sta buffer+oVolEntryLength
00A4FB  1  A9 0D         lda #kEntriesPerBlock
00A4FD  1  8D 24 8A      sta buffer+oVolEntriesPerBlock
00A500  1  A9 06         lda #kCanonicalFirstBitmapBlock
00A502  1  8D 27 8A      sta buffer+oVolBitmapNumber
00A505  1  85 0F         sta BitmapBase
00A507  1  A5 14         lda TotalBlocks+1
00A509  1  A6 13         ldx TotalBlocks
00A50B  1  8D 2A 8A      sta buffer+oVolTotalBlocks+1
00A50E  1  8E 29 8A      stx buffer+oVolTotalBlocks
00A511  1
00A511  1
00A511  1  A9 00         lda #>kRootDirectoryBlock
00A513  1  A2 02         ldx #<kRootDirectoryBlock
00A515  1  20 5F A7      jsr WriteBlockAX
00A518  1  B0 A3         bcs @exit
00A51A  1
00A51A  1              ; Construct and write blocks 3, 4, 5 (just links to the other blocks).
00A51A  1              @nextBlock:
00A51A  1  20 D0 A5      jsr ZeroBuffer
00A51D  1  E6 46         inc pdBlockNumberLow
00A51F  1  A6 46         ldx pdBlockNumberLow
00A521  1  E0 06         cpx #kCanonicalFirstBitmapBlock
00A523  1  B0 13         bcs @finishedCatalog
00A525  1  CA            dex
00A526  1  8E 00 8A      stx buffer+oDirLinkPrevious
00A529  1  E8            inx
00A52A  1  E8            inx
00A52B  1  E0 06         cpx #kCanonicalFirstBitmapBlock
00A52D  1  B0 03         bcs @noNextLink
00A52F  1  8E 02 8A      stx buffer+oDirLinkNext
00A532  1              @noNextLink:
00A532  1  20 63 A7      jsr DoWriteBlock
00A535  1  90 E3         bcc @nextBlock
00A537  1  60            rts
00A538  1              @finishedCatalog:
00A538  1
00A538  1              ; Construct and write the volume bitmap.
00A538  1              ;
00A538  1              ; Number of blocks is ceiling(TotalBlocks / 4096), which is the same as
00A538  1              ; TotalBlocks/256/ 16, plus 1 if there are any extra blocks (mod 4096).
00A538  1              ;
00A538  1  A5 14         lda TotalBlocks+1
00A53A  1  4A            lsr a
00A53B  1  4A            lsr a
00A53C  1  4A            lsr a
00A53D  1  4A            lsr a
00A53E  1  18            clc
00A53F  1  69 06         adc #kCanonicalFirstBitmapBlock
00A541  1  AA            tax
00A542  1  A5 14         lda TotalBlocks+1
00A544  1  29 0F         and #$0F
00A546  1  05 13         ora TotalBlocks
00A548  1  F0 01         beq @noExtra
00A54A  1  E8            inx
00A54B  1              @noExtra:
00A54B  1  86 12         stx PastLastBitmapBlock
00A54D  1
00A54D  1              @nextBitmapBlock:
00A54D  1  A6 46         ldx pdBlockNumberLow
00A54F  1  E8            inx
00A550  1  E4 12         cpx PastLastBitmapBlock
```

```
00A552  1  D0 06           bne @fullOfFreeBlocks
00A554  1  20 87 A5        jsr PopulateFinalBitmapBlock
00A557  1  4C 67 A5        jmp @writeBitmapBlock
00A55A  1
00A55A  1                @fullOfFreeBlocks:
00A55A  1  A9 FF           lda #$ff
00A55C  1  A2 00           ldx #0
00A55E  1                @allFree:
00A55E  1  9D 00 8A        sta buffer,x
00A561  1  9D 00 8B        sta buffer+256,x
00A564  1  CA              dex
00A565  1  D0 F7           bne @allFree
00A567  1
00A567  1                @writeBitmapBlock:
00A567  1  20 63 A7        jsr DoWriteBlock
00A56A  1  B0 1A           bcs @exit2
00A56C  1
00A56C  1  E6 46           inc pdBlockNumberLow
00A56E  1  A5 46           lda pdBlockNumberLow
00A570  1  C5 12           cmp PastLastBitmapBlock
00A572  1  90 D9           bcc @nextBitmapBlock
00A574  1
00A574  1                ; Mark blocks 0..LastBitmapBlock as used.
00A574  1  20 27 9C        jsr ReadFirstBitmapBlock
00A577  1  B0 0D           bcs @exit2
00A579  1                @allocateTheEarlyBlocks:
00A579  1  20 5D 9C        jsr AllocateOneBlockAX
00A57C  1  B0 08           bcs @exit2
00A57E  1  E8              inx
00A57F  1  E4 12           cpx PastLastBitmapBlock
00A581  1  90 F6           bcc @allocateTheEarlyBlocks
00A583  1  4C 1A 9D        jmp FlushBitmap
00A586  1
00A586  1                @exit2:
00A586  1  60              rts
00A587  1
00A587  1                ;--------------------------------------------------------------------------
00A587  1                ; PopulateFinalBitmapBlock
00A587  1                ;
00A587  1                ; Input:  TotalBlocks, TotalBlocks+1
00A587  1                ;         pdIOBuffer points to buffer
00A587  1                ;
00A587  1                ; Result: Fills in buffer (512 bytes) by setting the first (TotalBlocks % 4096)
00A587  1                ;         bits to 1 (free), and the rest of the bits to 0.
00A587  1                ;
00A587  1                PopulateFinalBitmapBlock:
00A587  1  A5 45           lda pdIOBufferHigh
00A589  1  48              pha
00A58A  1  20 D0 A5        jsr ZeroBuffer
00A58D  1  A5 14           lda TotalBlocks+1
00A58F  1  29 08           and #$08
00A591  1  F0 0C           beq @lessThanHalf
00A593  1
00A593  1  A9 FF           lda #$ff
00A595  1  A2 00           ldx #0
00A597  1                @freeFirstHalf:
00A597  1  9D 00 8A        sta buffer,x
00A59A  1  CA              dex
00A59B  1  D0 FA           bne @freeFirstHalf
00A59D  1  E6 45           inc pdIOBufferHigh
00A59F  1
00A59F  1                @lessThanHalf:
00A59F  1  A5 14           lda TotalBlocks+1
00A5A1  1  4A              lsr a
00A5A2  1  85 1F           sta num+1
00A5A4  1  A5 13           lda TotalBlocks
00A5A6  1  6A              ror a
00A5A7  1  46 1F           lsr num+1
00A5A9  1  6A              ror a
00A5AA  1  46 1F           lsr num+1
00A5AC  1  6A              ror a
00A5AD  1  AA              tax
00A5AE  1  A9 FF           lda #$ff
00A5B0  1  A0 00           ldy #0
00A5B2  1                @free8:
00A5B2  1  91 44           sta (pdIOBufferLow),y
00A5B4  1  C8              iny
00A5B5  1  CA              dex
00A5B6  1  D0 FA           bne @free8
00A5B8  1
00A5B8  1                ; Mark as free the last left-over 1 to 7 blocks.
00A5B8  1  A5 13           lda TotalBlocks
00A5BA  1  29 07           and #7
00A5BC  1  F0 06           beq @noMore
00A5BE  1  AA              tax
00A5BF  1  BD C8 A5        lda BitmapNFreeBlocksMasks,x
00A5C2  1  91 44           sta (pdIOBufferLow),y
00A5C4  1                @noMore:
00A5C4  1
00A5C4  1  68              pla
00A5C5  1  85 45           sta pdIOBufferHigh
00A5C7  1  60              rts
00A5C8  1
00A5C8  1                BitmapNFreeBlocksMasks:
00A5C8  1  00              .byte %00000000
00A5C9  1  80              .byte %10000000
00A5CA  1  C0              .byte %11000000
00A5CB  1  E0              .byte %11100000
00A5CC  1  F0              .byte %11110000
```

```
00A5CD  1  F8              .byte %11111000
00A5CE  1  FC              .byte %11111100
00A5CF  1  FE              .byte %11111110
00A5D0  1
00A5D0  1                ;--------------------------------------------------------------------------------------------
00A5D0  1                ;--------------------------------------------------------------------------------------------
00A5D0  1                ; Utility routines
00A5D0  1                ;--------------------------------------------------------------------------------------------
00A5D0  1                ;--------------------------------------------------------------------------------------------
00A5D0  1
00A5D0  1                ;------------------------------------------------------------------------
00A5D0  1                ; ZeroBuffer
00A5D0  1                ;
00A5D0  1                ; Result:  512 bytes of 0 in buffer
00A5D0  1                ;          A = 0, X = 0
00A5D0  1                ;
00A5D0  1                ZeroBuffer:
00A5D0  1  A9 00           lda #0
00A5D2  1  AA              tax
00A5D3  1                @zeroIndex:
00A5D3  1  9D 00 8A        sta buffer,x
00A5D6  1  9D 00 8B        sta buffer+256,x
00A5D9  1  CA              dex
00A5DA  1  D0 F7           bne @zeroIndex
00A5DC  1  60              rts
00A5DD  1
00A5DD  1                ;------------------------------------------------------------------------
00A5DD  1                ; LocateDirEntryForNameXY
00A5DD  1                ;
00A5DD  1                ; Input:  XY = pointer to filename
00A5DD  1                ; Result: Like LocateDirEntryForFilename
00A5DD  1                ;         (Filename is preserved.)
00A5DD  1                ;
00A5DD  1                LocateDirEntryForNameXY:
00A5DD  1  A5 03           lda Filename+1
00A5DF  1  48              pha
00A5E0  1  A5 02           lda Filename
00A5E2  1  48              pha
00A5E3  1
00A5E3  1  86 03           stx Filename+1
00A5E5  1  84 02           sty Filename
00A5E7  1  20 F3 A5        jsr LocateDirEntryForFilename
00A5EA  1
00A5EA  1  AA              tax
00A5EB  1  68              pla
00A5EC  1  85 02           sta Filename
00A5EE  1  68              pla
00A5EF  1  85 03           sta Filename+1
00A5F1  1  8A              txa
00A5F2  1  60              rts
00A5F3  1
00A5F3  1                ;------------------------------------------------------------------------
00A5F3  1                ; LocateDirEntryForFilename
00A5F3  1                ;
00A5F3  1                ; Input:   Filename
00A5F3  1                ; Result:  CLC, EntryPtr is set, DirectoryBlock is set
00A5F3  1                ;          SEC, A = error (typically eFileNotFound, eBadPathSyntax)
00A5F3  1                ;
00A5F3  1                API_FindDirEntry:
00A5F3  1                LocateDirEntryForFilename:
00A5F3  1  20 C7 A6        jsr SyntaxCheckFilename
00A5F6  1  B0 1F           bcs @exit
00A5F8  1
00A5F8  1  20 9B 9A        jsr OpenDir
00A5FB  1  B0 1A           bcs @exit
00A5FD  1                @next:
00A5FD  1  20 5D 9B        jsr ReadDir
00A600  1  B0 15           bcs @exit
00A602  1
00A602  1  A0 00           ldy #0
00A604  1  B1 0B           lda (EntryPtr),y
00A606  1  29 0F           and #$0f            ; strip off the storage type
00A608  1  D1 02           cmp (Filename),y    ; correct length?
00A60A  1  D0 F1           bne @next
00A60C  1  A8              tay
00A60D  1                @checkName:
00A60D  1  B1 0B           lda (EntryPtr),y
00A60F  1  D1 02           cmp (Filename),y
00A611  1  D0 EA           bne @next
00A613  1  88              dey
00A614  1  D0 F7           bne @checkName
00A616  1  18              clc
00A617  1                @exit:
00A617  1  60              rts
00A618  1
00A618  1                ;------------------------------------------------------------------------
00A618  1                ; LocateAvailableDirEntry
00A618  1                ;
00A618  1                ; Result:  CLC, EntryPtr is set, DirectoryBlock is set
00A618  1                ;          SEC, A = error
00A618  1                ;
00A618  1                ; Destroys "buffer" if we have to add a block to the directory.
00A618  1                ;
00A618  1                LocateAvailableDirEntry:
00A618  1  20 9B 9A        jsr OpenDir
00A61B  1  B0 09           bcs @error
00A61D  1                @next:
00A61D  1  20 65 9B        jsr ReadDir0
00A620  1  B0 04           bcs @error
```

```
00A622  1  D0 F9           bne @next
00A624  1  18              clc
00A625  1  60              rts
00A626  1
00A626  1                @error:
00A626  1  C9 46           cmp #eFileNotFound
00A628  1  D0 0E           bne @otherError
00A62A  1
00A62A  1  AD AE 87        lda FirstDirectoryBlock+1
00A62D  1  D0 0B           bne @extendSubdir
00A62F  1  AD AD 87        lda FirstDirectoryBlock
00A632  1  C9 02           cmp #kRootDirectoryBlock
00A634  1  D0 04           bne @extendSubdir
00A636  1
00A636  1  A9 49           lda #eDirectoryFull
00A638  1                @otherError:
00A638  1  38              sec
00A639  1                @exit:
00A639  1  60              rts
00A63A  1
00A63A  1                @extendSubdir:
00A63A  1  20 D0 A5        jsr ZeroBuffer
00A63D  1
00A63D  1  20 27 9C        jsr ReadFirstBitmapBlock
00A640  1  B0 F7           bcs @exit
00A642  1  20 5D 9C        jsr AllocateOneBlockAX
00A645  1  B0 F2           bcs @exit
00A647  1  8D 03 8C        sta DirectoryBuffer+oDirLinkNext+1
00A64A  1  8E 02 8C        stx DirectoryBuffer+oDirLinkNext
00A64D  1
00A64D  1  20 1A 9D        jsr FlushBitmap
00A650  1  B0 E7           bcs @exit
00A652  1
00A652  1                ; write the new block, with a prev-link to the old last block
00A652  1  A5 0E           lda DirectoryBlock+1
00A654  1  A6 0D           ldx DirectoryBlock
00A656  1  8D 01 8A        sta buffer+oDirLinkPrevious+1
00A659  1  8E 00 8A        stx buffer+oDirLinkPrevious
00A65C  1
00A65C  1  20 19 A7        jsr UseBuffer
00A65F  1  AD 03 8C        lda DirectoryBuffer+oDirLinkNext+1
00A662  1  AE 02 8C        ldx DirectoryBuffer+oDirLinkNext
00A665  1  20 5F A7        jsr WriteBlockAX
00A668  1  B0 CF           bcs @exit
00A66A  1
00A66A  1                ; update what used to be the last block, with a next-link to the new block
00A66A  1  20 58 A7        jsr WriteCurrentDirectoryBlock
00A66D  1  B0 CA           bcs @exit
00A66F  1
00A66F  1                ; find parent directory entry -- update block count, EOF
00A66F  1  20 69 A7        jsr FlushAndReadFirstDirectoryBlock
00A672  1  B0 C5           bcs @exit
00A674  1
00A674  1  AD 04 8C        lda DirectoryBuffer+oVolStorageType
00A677  1  29 F0           and #kStorageTypeMask
00A679  1  C9 E0           cmp #kSubdirHeader
00A67B  1  D0 39           bne @fileFormatError
00A67D  1
00A67D  1  AD 29 8C        lda DirectoryBuffer+oSubdirParentEntryNum
00A680  1  F0 34           beq @fileFormatError
00A682  1  C9 0E           cmp #kEntriesPerBlock+1
00A684  1  B0 30           bcs @fileFormatError
00A686  1  85 19           sta EntryCounter
00A688  1
00A688  1  AD 28 8C        lda DirectoryBuffer+oSubdirParentPointer+1
00A68B  1  AE 27 8C        ldx DirectoryBuffer+oSubdirParentPointer
00A68E  1  20 89 A7        jsr ReadDirectoryBlockAX
00A691  1  B0 A6           bcs @exit
00A693  1
00A693  1  20 92 9B        jsr InitEntryPtr
00A696  1                @loop:
00A696  1  C6 19           dec EntryCounter
00A698  1  F0 06           beq @done
00A69A  1  20 9B 9B        jsr BumpEntryPtr
00A69D  1  4C 96 A6        jmp @loop
00A6A0  1                @done:
00A6A0  1
00A6A0  1  A0 13           ldy #oBlockCount
00A6A2  1  A9 01           lda #1
00A6A4  1  20 BA A6        jsr AddToTwoByteEntryPtrField
00A6A7  1
00A6A7  1  A0 16           ldy #oFileSize+1
00A6A9  1  A9 02           lda #2
00A6AB  1  20 BA A6        jsr AddToTwoByteEntryPtrField
00A6AE  1
00A6AE  1  20 58 A7        jsr WriteCurrentDirectoryBlock
00A6B1  1  B0 86           bcs @exit
00A6B3  1
00A6B3  1                ; Start over from the beginning of the directory (inefficient, but easy)
00A6B3  1  4C 18 A6        jmp LocateAvailableDirEntry
00A6B6  1
00A6B6  1                @fileFormatError:
00A6B6  1  A9 4A           lda #eFileFormat
00A6B8  1  38              sec
00A6B9  1  60              rts
00A6BA  1
00A6BA  1                ;------------------------------------------------------------------------------
00A6BA  1                ; AddToTwoByteEntryPtrField
00A6BA  1                ;
```

```
00A6BA  1              ; Input:   A = value to add
00A6BA  1              ;          Y = offset into (EntryPtr)
00A6BA  1              ;
00A6BA  1              ; Destroyes:  A, Y.
00A6BA  1              ;
00A6BA  1              AddToTwoByteEntryPtrField:
00A6BA  1  18             clc
00A6BB  1  71 0B          adc (EntryPtr),y
00A6BD  1  91 0B          sta (EntryPtr),y
00A6BF  1  C8             iny
00A6C0  1  B1 0B          lda (EntryPtr),y
00A6C2  1  69 00          adc #0
00A6C4  1  91 0B          sta (EntryPtr),y
00A6C6  1  60             rts
00A6C7  1
00A6C7  1
00A6C7  1              ;--------------------------------------------------------------------------
00A6C7  1              ; SyntaxCheckFilename
00A6C7  1              ;
00A6C7  1              ; Input:   Filename
00A6C7  1              ; Result:  CLC, any lowercase letters in Filename are changed to uppercase
00A6C7  1              ;          SEC, A = error
00A6C7  1              ;
00A6C7  1              ; A valid filename has a length from 1 to 15.  The first character must be A-Z,
00A6C7  1              ; and the following characters must be A-Z, 0-9, or period.
00A6C7  1              ;
00A6C7  1              SyntaxCheckFilename:
00A6C7  1  A0 00          ldy #0
00A6C9  1  B1 02          lda (Filename),y    ; correct length?
00A6CB  1  F0 32          beq @syntax_bad
00A6CD  1  C9 10          cmp #16
00A6CF  1  B0 2E          bcs @syntax_bad
00A6D1  1  A8             tay
00A6D2  1              @char:
00A6D2  1  B1 02          lda (Filename),y
00A6D4  1
00A6D4  1  C9 7B          cmp #'z'+1
00A6D6  1  B0 27          bcs @syntax_bad
00A6D8  1  C9 61          cmp #'a'
00A6DA  1  90 06          bcc @not_lowercase
00A6DC  1  29 DF          and #%11011111
00A6DE  1  91 02          sta (Filename),y
00A6E0  1  D0 18          bne @next
00A6E2  1              @not_lowercase:
00A6E2  1
00A6E2  1  C9 5B          cmp #'Z'+1
00A6E4  1  B0 19          bcs @syntax_bad
00A6E6  1  C9 41          cmp #'A'
00A6E8  1  B0 10          bcs @next
00A6EA  1
00A6EA  1  C0 01          cpy #1
00A6EC  1  F0 11          beq @syntax_bad      ; first character must be a letter
00A6EE  1
00A6EE  1  C9 3A          cmp #'9'+1
00A6F0  1  B0 0D          bcs @syntax_bad
00A6F2  1  C9 30          cmp #'0'
00A6F4  1  B0 04          bcs @next
00A6F6  1
00A6F6  1  C9 2E          cmp #'.'
00A6F8  1  D0 05          bne @syntax_bad
00A6FA  1
00A6FA  1              @next:
00A6FA  1  88             dey
00A6FB  1  D0 D5          bne @char
00A6FD  1  18             clc
00A6FE  1  60             rts
00A6FF  1
00A6FF  1              @syntax_bad:
00A6FF  1  A9 40          lda #eBadPathSyntax
00A701  1  38             sec
00A702  1  60             rts
00A703  1
00A703  1              ;--------------------------------------------------------------------------
00A703  1              ; BlocksNeededFromFileSize
00A703  1              ;
00A703  1              ; Input:  FileSize (2 bytes)
00A703  1              ; Result: A = Number of disk blocks needed
00A703  1              ;
00A703  1              ; Destroys X.
00A703  1              ;
00A703  1              ; Compute  max(1, ceiling(FileSize / 512)), and then
00A703  1              ; add one more for a sapling file's index block.
00A703  1              ;
00A703  1              BlocksNeededFromFileSize:
00A703  1  A5 0A          lda FileSize+1
00A705  1  4A             lsr a
00A706  1  AA             tax                     ; X = FileSize / 512
00A707  1  A9 00          lda #0
00A709  1  6A             ror a                   ; A is nonzero if FileSize+1 was odd
00A70A  1  05 09          ora FileSize
00A70C  1  F0 01          beq @noExtraBytes
00A70E  1  E8             inx                     ; add 1 more block if there was a remainder
00A70F  1              @noExtraBytes:
00A70F  1  E0 02          cpx #2
00A711  1  90 03          bcc @needsJustOneBlock  ; 0 or 1 -> 1
00A713  1  E8             inx                     ; 1 more for the index block
00A714  1  8A             txa
00A715  1  60             rts
00A716  1
```

```
00A716  1              @needsJustOneBlock:
00A716  1  A9 01          lda #1
00A718  1  60             rts
00A719  1
00A719  1
00A719  1              ;----------------------------------------------------------------------------
00A719  1              UseBuffer:
00A719  1  A0 8A          ldy #>buffer
00A71B  1  D0 06          bne ioBufferY
00A71D  1              ;----------------------------------------------------------------------------
00A71D  1              UseDirectoryBuffer:
00A71D  1  A0 8C          ldy #>DirectoryBuffer
00A71F  1  D0 02          bne ioBufferY
00A721  1              ;----------------------------------------------------------------------------
00A721  1              ; Sets pdIOBuffer.  Preserves A,X and returns with Y=0.
00A721  1              ;
00A721  1              UseBitmapBuffer:
00A721  1  A0 8E          ldy #>BitmapBuffer
00A723  1              ioBufferY:
00A723  1  84 45          sty pdIOBufferHigh
00A725  1  A0 00          ldy #0
00A727  1  84 44          sty pdIOBufferLow
00A729  1  60             rts
00A72A  1
00A72A  1              ;----------------------------------------------------------------------------
00A72A  1              UseDestinationOrSpecialBuffer:
00A72A  1  AC 02 87       ldy SpecialFirstBlock
00A72D  1  F0 07          beq UseDestination
00A72F  1  20 23 A7       jsr ioBufferY
00A732  1  8C 02 87       sty SpecialFirstBlock
00A735  1  60             rts
00A736  1              UseDestination:
00A736  1  A4 01          ldy Destination+1
00A738  1  84 45          sty pdIOBufferHigh
00A73A  1  A4 00          ldy Destination
00A73C  1  84 44          sty pdIOBufferLow
00A73E  1  60             rts
00A73F  1
00A73F  1              ;----------------------------------------------------------------------------
00A73F  1              ; ReadBlockAXOrZeroes
00A73F  1              ;
00A73F  1              ; Read any block except #0.  If asked to read block 0, fill the buffer with
00A73F  1              ; 512 bytes of $00 instead.  This facilitates reading a sparse file, where
00A73F  1              ; the index block entry for a missing portion of a file is 0.
00A73F  1              ;
00A73F  1              ReadBlockAXOrZeroes:
00A73F  1  85 47          sta   pdBlockNumberHigh
00A741  1  86 46          stx   pdBlockNumberLow
00A743  1  05 46          ora   pdBlockNumberLow
00A745  1  D0 4D          bne   DoReadBlock
00A747  1  A8             tay
00A748  1              @zero1:
00A748  1  91 44          sta   (pdIOBufferLow),y
00A74A  1  C8             iny
00A74B  1  D0 FB          bne   @zero1
00A74D  1  E6 45          inc   pdIOBufferHigh
00A74F  1              @zero2:
00A74F  1  91 44          sta   (pdIOBufferLow),y
00A751  1  C8             iny
00A752  1  D0 FB          bne   @zero2
00A754  1  C6 45          dec   pdIOBufferHigh
00A756  1  18             clc
00A757  1  60             rts
00A758  1
00A758  1              ;----------------------------------------------------------------------------
00A758  1              WriteCurrentDirectoryBlock:
00A758  1  20 1D A7       jsr UseDirectoryBuffer
00A75B  1  A5 0E          lda DirectoryBlock+1
00A75D  1  A6 0D          ldx DirectoryBlock
00A75F  1              WriteBlockAX:
00A75F  1  85 47          sta pdBlockNumberHigh
00A761  1  86 46          stx pdBlockNumberLow
00A763  1              DoWriteBlock:
00A763  1  A9 02          lda #PRODOS_WRITE
00A765  1  85 42          sta pdCommandCode
00A767  1  D0 2F          bne CFBlockDriver
00A769  1
00A769  1              ;----------------------------------------------------------------------------
00A769  1              ; FlushAndReadFirstDirectoryBlock, FlushAndReadDirectoryBlockAX
00A769  1              ;
00A769  1              ; Read a directory block into DirectoryBuffer, writing out the current block
00A769  1              ; first.
00A769  1              ;
00A769  1              ; Result:  CLC for success
00A769  1              ;          SEC, A = error
00A769  1              ;
00A769  1              FlushAndReadFirstDirectoryBlock:
00A769  1  AD AE 87       lda FirstDirectoryBlock+1
00A76C  1  AE AD 87       ldx FirstDirectoryBlock
00A76F  1              FlushAndReadDirectoryBlockAX:
00A76F  1  C5 0E          cmp DirectoryBlock+1
00A771  1  D0 06          bne @different
00A773  1  E4 0D          cpx DirectoryBlock
00A775  1  D0 02          bne @different
00A777  1  18             clc
00A778  1  60             rts
00A779  1              @different:
00A779  1  48             pha
00A77A  1  8A             txa
```

```
00A77B  1  48              pha
00A77C  1  20 58 A7        jsr WriteCurrentDirectoryBlock
00A77F  1  90 05           bcc @noError
00A781  1  AA              tax
00A782  1  68              pla
00A783  1  68              pla
00A784  1  8A              txa
00A785  1  60              rts
00A786  1                @noError:
00A786  1  68              pla
00A787  1  AA              tax
00A788  1  68              pla
00A789  1                ReadDirectoryBlockAX:
00A789  1  20 1D A7        jsr   UseDirectoryBuffer
00A78C  1  85 0E           sta   DirectoryBlock+1
00A78E  1  86 0D           stx   DirectoryBlock
00A790  1                ReadBlockAX:
00A790  1  85 47           sta   pdBlockNumberHigh
00A792  1  86 46           stx   pdBlockNumberLow
00A794  1                DoReadBlock:
00A794  1  A9 01           lda   #PRODOS_READ
00A796  1  85 42           sta   pdCommandCode
00A798  1                ; Fall into CF block driver below.  If Block driver moves, place "jmp CFBlockDriver" here
00A798  1                ;------------------------------------------------------------------------------
00A798  1                ; Low Level CF Driver Entry point
00A798  1                ;
00A798  1                ; unit number not used currently
00A798  1                CFBlockDriver:
00A798  1  2C 01 87        bit CFFA1_Options
00A79B  1  10 09           bpl CFBlockDriverInternal
00A79D  1  20 64 A9        jsr LogBlockOperationBefore
00A7A0  1  20 A6 A7        jsr CFBlockDriverInternal
00A7A3  1  4C BC A9        jmp LogBlockOperationAfter
00A7A6  1
00A7A6  1                CFBlockDriverInternal:
00A7A6  1                 .if APPLE2
00A7A6  1                   jmp ReadOrWriteBlockAppleII
00A7A6  1                 .else
00A7A6  1                ;
00A7A6  1                ; Set CF card into 8bit mode
00A7A6  1                ;
00A7A6  1  A9 01           lda  #Enable8BitTransfers
00A7A8  1  8D F9 AF        sta  ATAFeature
00A7AB  1  A9 00           lda  #$00                      ; Drive=0
00A7AD  1  8D FE AF        sta  ATA_LBA27_24              ; Talk to the Master device and use LBA mode.
00A7B0  1  A9 EF           lda  #ATASetFeature
00A7B2  1  8D FF AF        sta  ATACommand                ;Issue the 8-bit feature command to the drive
00A7B5  1
00A7B5  1                ; Don't wait here for CF card to be ready. Ready is check below already.
00A7B5  1                ;;;;   jsr  IDEWaitReady               ;Wait for BUSY flag to clear
00A7B5  1
00A7B5  1
00A7B5  1                ; process the command code and jump to appropriate routine.
00A7B5  1
00A7B5  1  A5 42           lda  pdCommandCode
00A7B7  1  C9 01           cmp  #PRODOS_READ
00A7B9  1  F0 0F           beq  ReadBlock
00A7BB  1
00A7BB  1  C9 02           cmp  #PRODOS_WRITE
00A7BD  1  F0 63           beq  WriteBlock
00A7BF  1
00A7BF  1  C9 00           cmp  #PRODOS_STATUS
00A7C1  1  D0 03           bne  @notStatus
00A7C3  1  4C 9A A8        jmp  GetStatus
00A7C6  1                @notStatus:
00A7C6  1
00A7C6  1  A9 27           lda  #PRODOS_IO_ERROR
00A7C8  1  38              sec
00A7C9  1  60              rts
00A7CA  1
00A7CA  1                ;------------------------------------------------------------------------------
00A7CA  1                ; ReadBlock - Read a block from device into memory
00A7CA  1                ;
00A7CA  1                ; Input:
00A7CA  1                ;       pd Command Block Data $42 - $47
00A7CA  1                ;
00A7CA  1                ; Output:
00A7CA  1                ;       A = ProDOS read return code
00A7CA  1                ;       Carry flag: 0 = Okay, 1 = Error
00A7CA  1                ;
00A7CA  1                ; ZeroPage Usage:
00A7CA  1                ;       $F0
00A7CA  1                ;       Note: location $F0 is saved and restored before driver exits
00A7CA  1                ;
00A7CA  1                ReadBlock:
00A7CA  1
00A7CA  1  A5 45           lda  pdIOBufferHigh
00A7CC  1  48              pha
00A7CD  1  A5 F0           lda  zpt1
00A7CF  1  48              pha
00A7D0  1
00A7D0  1  20 DC A7        jsr ReadBlockCore
00A7D3  1  AA              tax                            ; ver1.3: Save Acc
00A7D4  1
00A7D4  1  68              pla
00A7D5  1  85 F0           sta  zpt1
00A7D7  1  68              pla
00A7D8  1  85 45           sta  pdIOBufferHigh
00A7DA  1  8A              txa                            ; ver1.3: restore Acc
```

```
00A7DB  1  60             rts
00A7DC  1
00A7DC  1              ReadBlockCore:
00A7DC  1
00A7DC  1  20 28 A9       jsr  IDEWaitReady
00A7DF  1  20 80 A8       jsr  SetupTaskFile          ;Program the device's task file registers
00A7E2  1                                             ; based on ProDOS address
00A7E2  1
00A7E2  1  A9 20          lda  #ATACRead
00A7E4  1  8D FF AF       sta  ATACommand             ;Issue the read command to the drive
00A7E7  1  20 28 A9       jsr  IDEWaitReady           ;Wait for BUSY flag to clear
00A7EA  1
00A7EA  1  AD FF AF       lda  ATAStatus              ;Check for error response from device
00A7ED  1  29 09          and  #$09
00A7EF  1  C9 01          cmp  #$01                   ;If DRQ=0 and ERR=1 a device error occured
00A7F1  1  D0 04          bne  rCommandOK
00A7F3  1
00A7F3  1              ;
00A7F3  1              ; The drive has returned an error code. Just return I/O error code to PRODOS
00A7F3  1              ;
00A7F3  1  A9 27          lda  #PRODOS_IO_ERROR
00A7F5  1  38             sec
00A7F6  1  60             rts
00A7F7  1              ;
00A7F7  1              ; Sector is ready to read
00A7F7  1              ;
00A7F7  1              rCommandOK:
00A7F7  1  A0 02          ldy  #2
00A7F9  1  84 F0          sty  zpt1
00A7FB  1  A0 00          ldy  #0
00A7FD  1
00A7FD  1              rLoop:
00A7FD  1  AD FF AF       lda  ATAStatus              ;Note: not using IDEWaitReady, using inline code
00A800  1  30 FB          bmi  rLoop                  ;Wait for BUSY (bit 7) to be zero
00A802  1  29 08          and  #$08                   ;get DRQ status bit
00A804  1  F0 18          beq  rShort                 ;if off, didn't get enough data
00A806  1
00A806  1  AD F8 AF       lda  ATAData
00A809  1  91 44          sta  (pdIOBufferLow),y
00A80B  1  C8             iny
00A80C  1
00A80C  1  AD F8 AF       lda  ATAData
00A80F  1  91 44          sta  (pdIOBufferLow),y
00A811  1  C8             iny
00A812  1
00A812  1  D0 E9          bne  rLoop
00A814  1  E6 45          inc  pdIOBufferHigh
00A816  1  C6 F0          dec  zpt1
00A818  1  D0 E3          bne  rLoop
00A81A  1
00A81A  1  A9 00          lda  #0
00A81C  1  18             clc
00A81D  1  60             rts
00A81E  1              ;
00A81E  1              ; The Block was short, return I/O error code to ProDOS
00A81E  1              ;
00A81E  1              rShort:
00A81E  1  A9 27          lda #PRODOS_IO_ERROR
00A820  1  38             sec
00A821  1  60             rts
00A822  1
00A822  1              ;---------------------------------------------------------------------
00A822  1              ; WriteBlock - Write a block in memory to device
00A822  1              ;
00A822  1              ; Input:
00A822  1              ;       pd Command Block Data $42 - $47
00A822  1              ;       X = requested slot number in form $n0 where n = slot 1 to 7
00A822  1              ;
00A822  1              ; Output:
00A822  1              ;       A = ProDOS write return code
00A822  1              ;       Carry flag: 0 = Okay, 1 = Error
00A822  1              ;
00A822  1              WriteBlock:
00A822  1  A5 45          lda  pdIOBufferHigh
00A824  1  48             pha
00A825  1  A5 F0          lda  zpt1
00A827  1  48             pha
00A828  1
00A828  1  20 34 A8       jsr  WriteBlockCore
00A82B  1  AA             tax
00A82C  1
00A82C  1  68             pla
00A82D  1  85 F0          sta  zpt1
00A82F  1  68             pla
00A830  1  85 45          sta  pdIOBufferHigh
00A832  1  8A             txa
00A833  1  60             rts
00A834  1
00A834  1              WriteBlockCore:
00A834  1  20 28 A9       jsr  IDEWaitReady
00A837  1  20 80 A8       jsr  SetupTaskFile          ;program IDE task file
00A83A  1
00A83A  1              ; Write sector from RAM
00A83A  1  A9 30          lda  #ATACWrite
00A83C  1  8D FF AF       sta  ATACommand
00A83F  1  20 28 A9       jsr  IDEWaitReady
00A842  1
00A842  1  AD FF AF       lda  ATAStatus              ;Check for error response from writing command
00A845  1  29 09          and  #$09
```

```
00A847  1  C9 01           cmp  #$01                    ;if DRQ=0 and ERR=1 an error occured
00A849  1  D0 04           bne  wCommandOK
00A84B  1
00A84B  1               ; The drive has returned an error code. Just return I/O error code to PRODOS
00A84B  1               ;
00A84B  1  A9 27           lda  #PRODOS_IO_ERROR
00A84D  1  38              sec
00A84E  1  60              rts
00A84F  1               ;
00A84F  1               ; Sector is ready to write
00A84F  1               ;
00A84F  1               wCommandOK:
00A84F  1  A0 02           ldy  #2
00A851  1  84 F0           sty  zpt1
00A853  1  A0 00           ldy  #0
00A855  1
00A855  1               wLoop:
00A855  1  AD FF AF        lda  ATAStatus               ;Note: not using IDEWaitReady, using inline code
00A858  1  30 FB           bmi  wLoop                   ;Wait for BUSY (bit 7) to be zero
00A85A  1  29 08           and  #$08                    ;get DRQ status bit
00A85C  1  F0 1E           beq  wShort                  ;if off, didn't get enough data
00A85E  1
00A85E  1  2C F1 AF        bit  SetCSMask               ;Block all read cycles (esp the extra ones preceeding write cycles)
00A861  1
00A861  1  B1 44           lda  (pdIOBufferLow),y
00A863  1  8D F8 AF        sta  ATAData
00A866  1  C8              iny
00A867  1  B1 44           lda  (pdIOBufferLow),y
00A869  1  8D F8 AF        sta  ATAData
00A86C  1
00A86C  1  2C F2 AF        bit  ClearCSMask             ;Set back to normal, allow CS0 assertions on read cycles
00A86F  1
00A86F  1  C8              iny
00A870  1  D0 E3           bne  wLoop
00A872  1  E6 45           inc  pdIOBufferHigh
00A874  1  C6 F0           dec  zpt1
00A876  1  D0 DD           bne  wLoop
00A878  1
00A878  1  A9 00           lda  #0
00A87A  1  18              clc
00A87B  1  60              rts
00A87C  1               ;
00A87C  1               ; The Block was short, return I/O error code to PRODOS
00A87C  1               ;
00A87C  1               wShort:
00A87C  1  A9 27           lda  #PRODOS_IO_ERROR
00A87E  1  38              sec
00A87F  1  60              rts
00A880  1
00A880  1               ;-------------------------------------------------------------------------------
00A880  1               ; SetupTaskFile - Program CF registers with block address to be accessed
00A880  1               ; For now, no block number to LBA translation
00A880  1               ;
00A880  1               SetupTaskFile:
00A880  1
00A880  1  A5 46           lda  pdBlockNumberLow
00A882  1  8D FB AF        sta  ATA_LBA07_00            ;store ProDOS Low block # into LBA 0-7
00A885  1
00A885  1  A5 47           lda  pdBlockNumberHigh
00A887  1  8D FC AF        sta  ATA_LBA15_08            ;store ProDOS High block # into LBA 15-8
00A88A  1
00A88A  1  A9 00           lda  #0
00A88C  1  8D FD AF        sta  ATA_LBA23_16             ;store LBA bits 23-16
00A88F  1
00A88F  1  A9 E0           lda  #$E0                    ; 1, (LBA), 1, (Drive), LBA 27-24, where LBA=1, Drive=0
00A891  1  8D FE AF        sta  ATA_LBA27_24                 ; Talk to the Master device and use LBA mode.
00A894  1
00A894  1  A9 01           lda  #1
00A896  1  8D FA AF        sta  ATASectorCnt
00A899  1  60              rts
00A89A  1
00A89A  1               ;-------------------------------------------------------------------------------
00A89A  1               ; GetStatus - Called by ProDOS and SmartPort to get device status and size
00A89A  1               ;
00A89A  1               ; Input:
00A89A  1               ;       DriveNumber (0 to 3)
00A89A  1               ;
00A89A  1               ; Output:
00A89A  1               ;       A = ProDOS status return code
00A89A  1               ;       X = drive size LSB
00A89A  1               ;       Y = drive size MSB
00A89A  1               ;       Carry flag: 0 = Okay, 1 = Error
00A89A  1               ;       DrvBlkCount0..DrvBlkCount2 = usable blocks on device
00A89A  1               ;
00A89A  1               GetStatus:
00A89A  1               ;
00A89A  1               ; Determine if a drive/device is present.
00A89A  1               ;
00A89A  1  20 32 A9        jsr  CheckDevice
00A89D  1  B0 6F           bcs  NoDrive
00A89F  1
00A89F  1               ; Device is present
00A89F  1               sDriveOK:
00A89F  1  20 28 A9        jsr  IDEWaitReady
00A8A2  1  A9 EC           lda  #ATAIdentify
00A8A4  1  8D FF AF        sta  ATACommand              ;Issue the read command to the drive
00A8A7  1  20 28 A9        jsr  IDEWaitReady            ;Wait for BUSY flag to go away
00A8AA  1
00A8AA  1  AD FF AF        lda  ATAStatus               ;Check for error response
```

```
00A8AD  1  29 09           and  #$09
00A8AF  1  C9 01           cmp  #$01                    ;if DRQ=0 and ERR=1 an error occured
00A8B1  1  D0 04           bne  sValidATACommand
00A8B3  1
00A8B3  1  A9 27           lda  #PRODOS_IO_ERROR
00A8B5  1  D0 59           bne  sError                  ; Command Error occured, return error
00A8B7  1
00A8B7  1                sValidATACommand:
00A8B7  1  A0 00           ldy  #$00                    ;zero loop counter
00A8B9  1
00A8B9  1                sPrefetchloop:
00A8B9  1  20 28 A9        jsr  IDEWaitReady            ;See if a word is ready
00A8BC  1  B0 04           bcs  sWordRdy
00A8BE  1  A9 27           lda  #PRODOS_IO_ERROR
00A8C0  1  D0 4E           bne  sError
00A8C2  1
00A8C2  1                sWordRdy:
00A8C2  1  AD F8 AF        lda  ATAData                 ;Read words 0 thru 56 but throw them away
00A8C5  1  C8              iny
00A8C6  1  C0 72           cpy  #114   ;57 * 2          ;Number of the last byte you want to throw away
00A8C8  1  D0 EF           bne  sPrefetchloop
00A8CA  1
00A8CA  1                sPrefetchDone:
00A8CA  1  AD F8 AF        lda  ATAData                 ;Read the current capacity in sectors (LBA)
00A8CD  1                 .if BLOCKOFFSET<>0
00A8CD  1                   sec
00A8CD  1                   sbc  #BLOCKOFFSET
00A8CD  1                 .endif
00A8CD  1  8D 05 87         sta  DrvBlkCount0
00A8D0  1  AD F8 AF         lda  ATAData
00A8D3  1                 .if BLOCKOFFSET<>0
00A8D3  1                   sbc  #0
00A8D3  1                 .endif
00A8D3  1  8D 06 87         sta  DrvBlkCount1
00A8D6  1  AD F8 AF         lda  ATAData
00A8D9  1                 .if BLOCKOFFSET<>0
00A8D9  1                   sbc  #0
00A8D9  1                 .endif
00A8D9  1  8D 07 87         sta  DrvBlkCount2
00A8DC  1
00A8DC  1
00A8DC  1  AD 07 87         lda  DrvBlkCount2
00A8DF  1  C9 04            cmp  #PARTITIONS32MB         ;max out at (#PARTITIONS32MB * $10000 + 00FFFF)
00A8E1  1                                                ; blocks
00A8E1  1  B0 05            bcs  maxOutAtN
00A8E3  1
00A8E3  1  AD F8 AF         lda  ATAData
00A8E6  1  F0 0D            beq  lessThan8GB
00A8E8  1                maxOutAtN:
00A8E8  1  A9 FF            lda  #$FF                    ;The device is truly huge! Just set our 3-byte
00A8EA  1                                                ; block count to $03FFFF
00A8EA  1  8D 05 87         sta  DrvBlkCount0
00A8ED  1  8D 06 87         sta  DrvBlkCount1
00A8F0  1  A9 03            lda  #PARTITIONS32MB-1       ;Number of 32MB devices, set by the equate:
00A8F2  1                                                ; #PARTITIONS32MB
00A8F2  1  8D 07 87         sta  DrvBlkCount2
00A8F5  1                lessThan8GB:
00A8F5  1
00A8F5  1
00A8F5  1                PostFetch:
00A8F5  1  20 28 A9         jsr IDEWaitReady             ;read the rest of the words, until command ends
00A8F8  1  90 06            bcc sReadComplete
00A8FA  1  AD F8 AF         lda ATAData
00A8FD  1  4C F5 A8         jmp PostFetch
00A900  1                sReadComplete:
00A900  1
00A900  1                ; DrvBlkCount2 is the number of 32 MB partitions availiable - 1,
00A900  1                ; or the highest drive # supported (zero based).
00A900  1                ;
00A900  1                ; If DrvBlkCount2 > drive # then StatusSize = $FFFF
00A900  1                ; If DrvBlkCount2 = drive # then StatusSize = DrvBlkCount1,DrvBlkCount0
00A900  1                ; If DrvBlkCount2 < drive # then StatusSize = 0
00A900  1                ;
00A900  1                ; This scheme has a special case which must be handled because ProDOS
00A900  1                ;   partitions are not quite 32 meg in size but are only FFFF blocks in size.
00A900  1                ;   If a device is exactly: 32meg or 10000h blocks in size, it would appear
00A900  1                ;   as one drive of size FFFF and another drive of size 0000. To handle this
00A900  1                ;   case, we check for an exact size of 0000 and fall into the NoDrive code.
00A900  1                ;
00A900  1  AD 03 87         lda DriveNumber
00A903  1  CD 07 87         cmp DrvBlkCount2
00A906  1  F0 0E            beq ExactSize
00A908  1  A2 FF            ldx #$FF
00A90A  1  A0 FF            ldy #$FF
00A90C  1  90 16            bcc sNoError              ; full size ($FFFF blocks)
00A90E  1
00A90E  1                NoDrive:
00A90E  1  A9 2F            lda  #PRODOS_OFFLINE
00A910  1                sError:
00A910  1  A2 00            ldx  #0
00A912  1  A0 00            ldy  #0
00A914  1  38               sec
00A915  1  60               rts
00A916  1
00A916  1                ExactSize:                   ;If equal, then DrvBlkCount1,DrvBlkCount0 is the
00A916  1                                             ; drive's exact size
00A916  1  AD 05 87         lda  DrvBlkCount0
00A919  1  0D 06 87         ora  DrvBlkCount1
00A91C  1  F0 F0            beq  NoDrive              ;can't have a 0-block device
```

```
00A91E  1
00A91E  1  AE 05 87        ldx  DrvBlkCount0
00A921  1  AC 06 87        ldy  DrvBlkCount1
00A924  1               sNoError:
00A924  1  A9 00           lda  #0
00A926  1  18              clc                         ;no errors
00A927  1  60              rts
00A928  1
00A928  1               ;--------------------------------------------------------------------------
00A928  1               ; IDEWaitReady - Waits for BUSY flag to clear, and returns DRQ bit status
00A928  1               ;
00A928  1               ; Input:
00A928  1               ;       none
00A928  1               ; Ouput:
00A928  1               ;      Carry flag set if DRQ bit is on
00A928  1               ;
00A928  1               ; ZeroPage Usage:
00A928  1               ;       None
00A928  1               ;
00A928  1               ; CPU Registers changed:  A, P
00A928  1               ;
00A928  1               IDEWaitReady:
00A928  1  AD FF AF        lda  ATAStatus
00A92B  1  30 FB           bmi  IDEWaitReady           ;Wait for BUSY (bit 7) to be zero
00A92D  1  6A              ror                         ;shift DRQ status bit into the Carry bit
00A92E  1  6A              ror
00A92F  1  6A              ror
00A930  1  6A              ror
00A931  1  60              rts
00A932  1
00A932  1               ;--------------------------------------------------------------------------
00A932  1               ; CheckDevice - Check to see if a device is attached to the interface.
00A932  1               ;
00A932  1               ; Input:
00A932  1               ;       none
00A932  1               ; Output:
00A932  1               ;       Carry flag: 0 = Device Present, 1 = Device Missing
00A932  1               ;
00A932  1               ; CPU Registers changed:  A, P
00A932  1               ;
00A932  1               ;  Checks to see if the drive status register is readable and equal to $50
00A932  1               ;  If so, return with the Carry clear, otherwise return with the carry set.
00A932  1               ;  Waits up to 10sec on a standard 1MHz system for drive to become ready.
00A932  1               ;
00A932  1               CheckDevice:
00A932  1  98              tya
00A933  1  48              pha
00A934  1  2C F2 AF        bit  ClearCSMask            ;reset MASK bit in PLD for normal CS0 signaling
00A937  1  A9 E0           lda  #$E0                   ;$E0 = [1, LBA, 1, Drive, LBA 27-24]  where
00A939  1                                              ; LBA=1, Drive=0
00A939  1  8D FE AF        sta  ATA_LBA27_24               ;Make sure ATA master drive is accessed
00A93C  1
00A93C  1  A0 00           ldy  #0
00A93E  1               chkLoop:
00A93E  1  AD FF AF        lda  ATAStatus
00A941  1  29 D0           and  #%11010000
00A943  1  C9 50           cmp  #$50                   ;if BUSY= 0 and RDY=1 and DSC=1
00A945  1  F0 0D           beq  DeviceFound
00A947  1  A9 C5           lda  #WAIT_100ms
00A949  1  20 58 A9        jsr  Wait                   ;Wait 100ms for device to be ready
00A94C  1  C8              iny
00A94D  1  C0 64           cpy  #100                   ;Wait up to 10 seconds for drive to be ready
00A94F  1  D0 ED           bne chkLoop
00A951  1
00A951  1  38              sec                         ;set c = 1 if drive is not attached
00A952  1  B0 01           bcs  DeviceExit
00A954  1
00A954  1               DeviceFound:
00A954  1  18              clc                         ;set c = 0 if drive is attached
00A955  1
00A955  1               DeviceExit:
00A955  1  68              pla
00A956  1  A8              tay
00A957  1  60              rts
00A958  1                .endif
00A958  1
00A958  1               ;--------------------------------------------------------------------------
00A958  1               ;  Wait - Copy of Apple's wait routine. Can't use ROM based routine in case
00A958  1               ;         ROM is not active when we need it.
00A958  1               ;
00A958  1               ; Input:
00A958  1               ;       A = desired delay time, where Delay(us) = .5(5A^2 + 27A + 26)
00A958  1               ;       or more usefully: A = (Delay[in uS]/2.5 + 2.09)^.5 - 2.7
00A958  1               ;
00A958  1               ; CPU Registers changed:  A, P
00A958  1               ;
00A958  1               Wait:
00A958  1  38              sec
00A959  1
00A959  1               Wait2:
00A959  1  48              pha
00A95A  1
00A95A  1               Wait3:
00A95A  1  E9 01           sbc  #1
00A95C  1  D0 FC           bne  Wait3
00A95E  1  68              pla
00A95F  1  E9 01           sbc  #1
00A961  1  D0 F6           bne  Wait2
00A963  1  60              rts
```

```
00A964  1
00A964  1
00A964  1               .if APPLE2
00A964  1              ;--------------------------------------------------------------------------
00A964  1              ; ReadOrWriteBlockAppleII
00A964  1              ;
00A964  1              ; Input:  pdCommandCode, pdBlockNumber, pdIOBuffer
00A964  1              ; Result: CLC for success, SEC/A=error
00A964  1              ;
00A964  1              ReadOrWriteBlockAppleII:
00A964  1                 lda pdIOBufferHigh
00A964  1                 ldx pdIOBufferLow
00A964  1                 sta rw_buffer+1
00A964  1                 stx rw_buffer
00A964  1                 lda pdBlockNumberHigh
00A964  1                 ldx pdBlockNumberLow
00A964  1                 sta rw_block+1
00A964  1                 stx rw_block
00A964  1                 lda  pdCommandCode
00A964  1                 cmp  #PRODOS_READ
00A964  1                 beq  ReadBlockA2
00A964  1                 cmp  #PRODOS_WRITE
00A964  1                 beq  WriteBlockA2
00A964  1                 cmp  #PRODOS_STATUS
00A964  1                 beq  StatusA2
00A964  1                 lda  #PRODOS_IO_ERROR
00A964  1                 sec
00A964  1                 rts
00A964  1
00A964  1              ReadBlockA2:
00A964  1                 jsr $bf00
00A964  1                 .byte $80
00A964  1                 .word rw_parms
00A964  1                 rts
00A964  1
00A964  1              WriteBlockA2:
00A964  1                 jsr $bf00
00A964  1                 .byte $81
00A964  1                 .word rw_parms
00A964  1                 rts
00A964  1
00A964  1              rw_parms:
00A964  1                 .byte 3
00A964  1                 .byte $D0  ;slot 5, drive 2
00A964  1              rw_buffer:
00A964  1                 .word 0
00A964  1              rw_block:
00A964  1                 .word 0
00A964  1
00A964  1              StatusA2:
00A964  1                 lda DriveNumber
00A964  1                 cmp #2
00A964  1                 bcs @err
00A964  1                 cmp #1
00A964  1                 beq @status1
00A964  1              @status0:
00A964  1                 ldy #$12
00A964  1                 ldx #$34
00A964  1                 lda #0
00A964  1                 clc
00A964  1                 rts
00A964  1              @status1:
00A964  1                 ldy #$23
00A964  1                 ldx #$45
00A964  1                 lda #0
00A964  1                 clc
00A964  1                 rts
00A964  1              @err:
00A964  1                 lda #PRODOS_OFFLINE
00A964  1                 sec
00A964  1                 rts
00A964  1               .endif
00A964  1
00A964  1              ;--------------------------------------------------------------------------
00A964  1              ; LogBlockOperationBefore
00A964  1              ;
00A964  1              ; Input:  pdCommandCode, pdIOBuffer, pdBlockNumber
00A964  1              ; Result: Message displayed, such as:
00A964  1              ;     [READ  $0002 @ $7A00
00A964  1              ;     [WRITE $0006 @ $7E00
00A964  1              ;     [STATUS $00
00A964  1              ;
00A964  1              LogBlockOperationBefore:
00A964  1  A5 42           lda pdCommandCode
00A966  1  C9 01           cmp #PRODOS_READ
00A968  1  F0 12           beq @read
00A96A  1  C9 02           cmp #PRODOS_WRITE
00A96C  1  F0 1C           beq @write
00A96E  1  C9 00           cmp #PRODOS_STATUS
00A970  1  F0 38           beq @status
00A972  1  20 8C 96        jsr DispString
00A975  1  5B 3F 3F 3F     .byte "[??? ",0
00A979  1  20 00
00A97B  1  60              rts
00A97C  1              @read:
00A97C  1  20 8C 96        jsr DispString
00A97F  1  5B 52 45 41     .byte "[READ  ",0
00A983  1  44 20 20 00
00A987  1  4C 95 A9        jmp @common
```

```
00A98A  1              @write:
00A98A  1  20 8C 96       jsr DispString
00A98D  1  5B 57 52 49    .byte "[WRITE ",0
00A991  1  54 45 20 00
00A995  1              @common:
00A995  1  A5 47          lda pdBlockNumberHigh
00A997  1  A6 46          ldx pdBlockNumberLow
00A999  1  20 15 9A       jsr PrintHexAX
00A99C  1  20 8C 96       jsr DispString
00A99F  1  20 40 20 00    .byte " @ ",0
00A9A3  1  A5 45          lda pdIOBufferHigh
00A9A5  1  A6 44          ldx pdIOBufferLow
00A9A7  1  4C 15 9A       jmp PrintHexAX
00A9AA  1              @status:
00A9AA  1  20 8C 96       jsr DispString
00A9AD  1  5B 53 54 41    .byte "[STATUS ",0
00A9B1  1  54 55 53 20
00A9B5  1  00
00A9B6  1  AD 03 87       lda DriveNumber
00A9B9  1  4C 25 98       jmp DispByteWithDollarSign
00A9BC  1
00A9BC  1              ;------------------------------------------------------------------------------
00A9BC  1              ; LogBlockOperationAfter
00A9BC  1              ;
00A9BC  1              ; If CLC, just displays the "]",CR to end the log message.
00A9BC  1              ; If SEC, displays the error code in A as well.
00A9BC  1              ;
00A9BC  1              ; Preserves:  A, P.
00A9BC  1              ;
00A9BC  1              LogBlockOperationAfter:
00A9BC  1  08             php
00A9BD  1  48             pha
00A9BE  1  90 0D          bcc @noError
00A9C0  1  20 8C 96       jsr DispString
00A9C3  1  20 45 52 52    .byte " ERR=$",0
00A9C7  1  3D 24 00
00A9CA  1  20 2A 98       jsr DispByte
00A9CD  1              @noError:
00A9CD  1  20 8C 96       jsr DispString
00A9D0  1  5D 8D 00       .byte "]",CR,0
00A9D3  1  68             pla
00A9D4  1  28             plp
00A9D5  1  60             rts
00A9D6  1
00A9D6  1
00A9D6  1              ;------------------------------------------------------------------------------
00A9D6  1              ; At the end of the 8K of ROM, put our ID bytes and a version number.
00A9D6  1              ;------------------------------------------------------------------------------
00A9D6  1  xx xx xx xx    .res Origin+$2000-32-4-*
00A9DA  1  xx xx xx xx

…

00AFDC  1  CF FA          .byte $CF,$FA           ; $AFFC, $AFFD = ID bytes
00AFDE  1  01             .byte OLDEST_COMPAT_VER  ; $AFFE = oldest API-compatible version
00AFDF  1  01             .byte FIRMWARE_VER       ; $AFFF = version
00AFE0  1              ;------------------------------------------------------------------------------
00AFE0  1              ; END
00AFE0  1              ;------------------------------------------------------------------------------
00AFE0  1
00AFE0  1
```