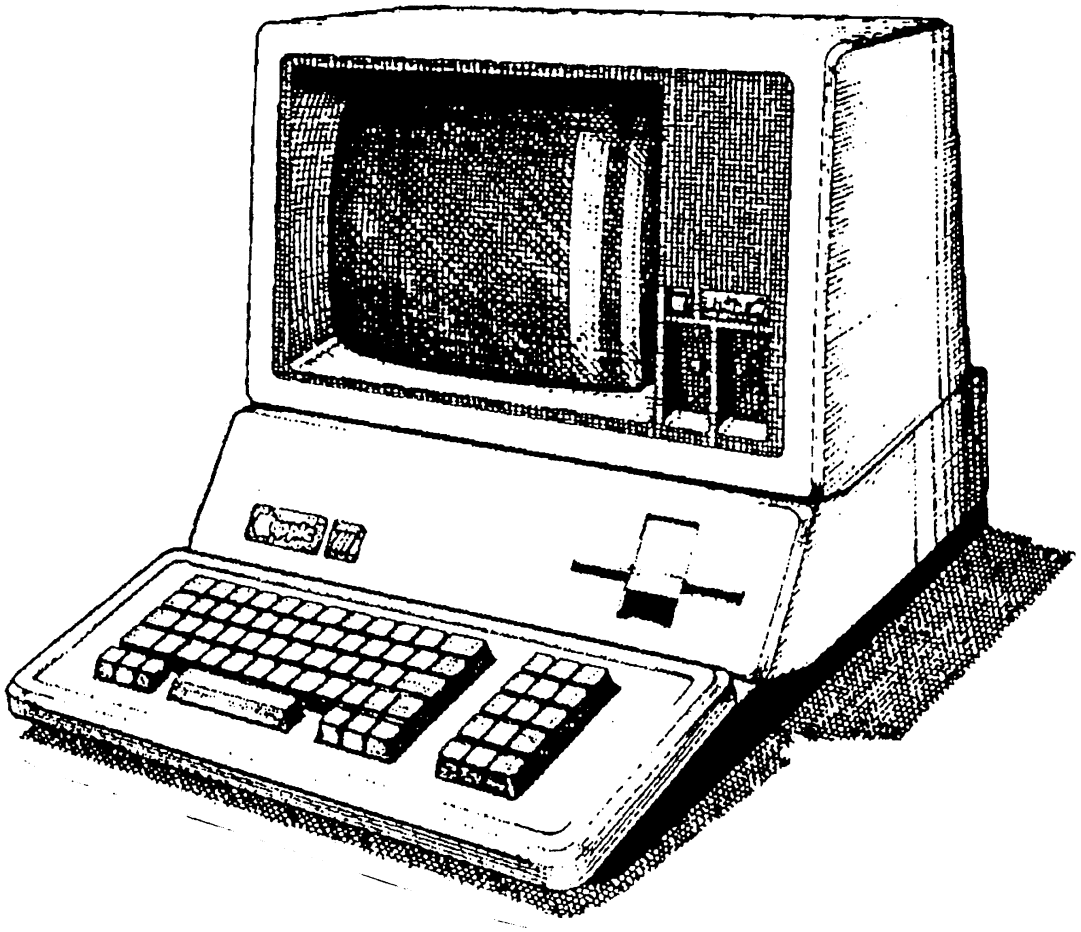
SEE DOC # 85, 79

Apple /// Computer Information



DOCUMENT NAME

INSIDE THE APPLE /// COMPUTER ROM

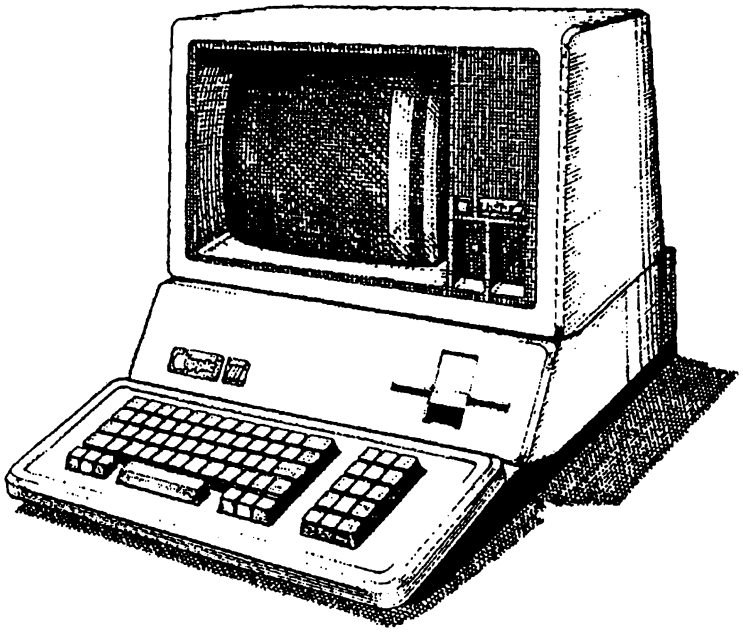
#

193

Ex Libris David T. Craig

"_174.PICT" 758 KB 2001-08-13 dpi: 600h x 600v pix: 4331h x 6083v

Apple III Computer Information



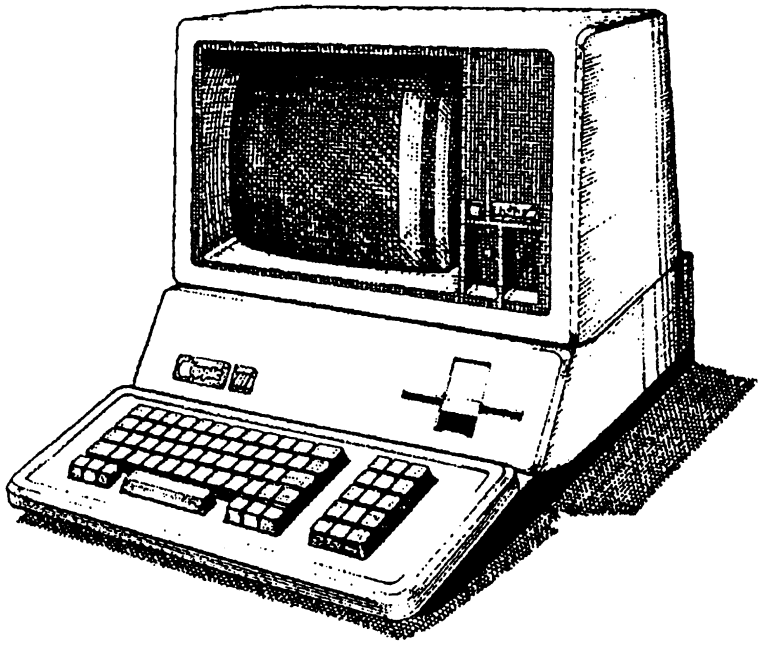
Inside the Apple III ROM

Document Table of Contents

Revision 2 • 04 Dec 1997
Revision 1 • 30 Nov 1997

"_175.PICT" 368 KB 2001-08-13 dpi: 600h x 600v pix: 3882h x 5480v

Apple III Computer Information



Inside the Apple III ROM

Revision 2 • 04 Dec 1997

"_176.PICT" 328 KB 2001-08-13 dpi: 600h x 600v pix: 4308h x 5030v

Inside the Apple /// Computer ROM

David T. Craig • 04 December 1997
71533.606@compuserve.com

TABLE OF CONTENTS

1 INTRODUCTION

This document provides a general overview of the contents of the Apple /// computer ROM revision 1. This information should be used in conjunction with a copy of the ROM source code listing. The audience of this document is anyone with an interest in the technology of the Apple /// computer's hardware and software.

NOTE

There were two revisions of the Apple /// ROM, revision 0 and revision 1. Revision 0 ROMs had at address F1B9 the value 60. Revision 1 ROMs had at address F1B9 the value A0.

This ROM contains 4 KB of 6502 programming and several data tables. The ROM occupies memory addresses F000-FFFF. The basic purpose of the ROM is to test the Apple /// computer hardware and boot an operating system from the ///'s built-in floppy disk drive. The ROM also contains a simple Monitor program whose purpose is to allow the user to interact with the /// at the hexadecimal level.

Apple planned from an architectural perspective to support two 4K ROMs. But only one ROM was ever created. The Environment Register let you control which ROM was active. Both ROMs shared the same address space so you could only have one ROM active at a time. This feature would have doubled the ROM's effective size providing Apple with more room for ROM-based features that higher-level /// software (e.g. SOS) could have used.

When the Apple /// computer is turned on the ROM's flow of execution is as follows:

1) The ROM starts execution at the address contained in FFFC-FFFD (RESET) which is address F4EE (DIAGN).

2) Diagnostics (DIAGN/F4EE) starts. The diagnostic first initializes some memory for the ROM's use. If the Open Apple and the Control keys are held down then enter the ROM Monitor. Otherwise run several diagnostic checks of the /// hardware (tests zero page, sizes memory, initializes screen text buffer, tests stack memory, tests ROM checksum, tests VIA chip, tests ACIA chip, tests A/D circuitry, tests keyboard connection). Any diagnostic failures display an error message and the user has to reset the computer.

3) Read block 0 (512 bytes) to address A000 from the floppy disk in the built-in disk drive (BOOT/F6A1). If no disk is found or block 0 cannot be read then display "RETRY" and wait for the user to reset the computer. If the block is successfully read then execute the block contents (this is called the SOS Bootstrap Loader: see section ROM USAGE BY SOS).

2 ROM SECTIONS

Section	Address	Purpose
Disk I/O	F000-F4C4	Read and write floppy disk blocks (512 bytes each)
Diagnostics	F4C5-F7FE	Diagnose the /// hardware
Monitor	F7FF-FFFF	Interacts with user so user can do simple things

3 IMPORTANT ROM ROUTINES

BLOCKIO / F479 — Reads or write a disk block (512 bytes), calls routine REGRWTS (F000) which reads a sector (256 bytes) from the disk.

BOOT / F6A1 — Read floppy disk block #0 into address A000, execute the block.

ENTRY / F901 — Monitor entry point.

DIAGN / F4EE — Diagnostic entry point.

USRENTRY/F6E6 — Tests RAM and displays a table showing chip failures (users may execute this routine from the Monitor). This test is aimed at Apple ///s with 128K of RAM that exists on the older 12-Volt RAM boards. Though this routine will work with the newer 5-Volt RAM boards (256K) this test shows wrong information when RAM errors occur since the two RAM boards contain a different number of RAM chips. You can identify the different RAM boards as follows: The 5V boards have a large gray ceramic resistor near the edge and the 12V boards have a small blue tubular capacitor. To test the ///'s RAM you really should use Apple's /// Diagnostics Disk which lets you specify which RAM board you have.

4 ROM TABLES

Here's a list of the important data tables in the ROM. This list does not include disk I/O tables.

Table Name / Address	Contents
CHRSET / FEC5-FFB3	Default character set (overridden when SOS loads the character set from SOS.DRIVER)
Copyright / FFC0-FFEF	Copyright message (contains the initials "JRH" for J. R. "Dick" Huston who was a key player behind the /// and SOS)
NMI / FFFA-FFFB	Jump address for the Non-Maskable Interrupt signal
RESET / FFFC-FFFD	Jump address when the /// is powered on
IRQ / FFFE-FFFF	Jump address for the Interrupt Request signal

5 ROM USAGE BY SOS

The Apple /// operating system (SOS = Sophisticated Operating System or Sara's OS) uses several ROM routines. These routines seem to all be related to disk block I/O. The following discussion is based on SOS version 1.3.

When the ROM loads block 0 from a SOS disk the ROM is loading the SOS Bootstrap Loader program. This program, which is at most 512 bytes in length, uses the ROM routine REGRWTS (F000) to read the SOS Loader into memory. This program does not test the ROM revision. It is interesting to note that ROM routine BLOCKIO is not used, instead a lower-level routine (REGRWTS) is used.

The SOS Loader determines if the ROM is revision 1 by comparing address F1B9's contents against A0 (reference: SOS source file SOSLDR.D.SRC). If this comparison fails then SOS displays on the screen the error "ROM ERROR: PLEASE NOTIFY YOUR DEALER." If the ROM revision is correct then the SOS loader uses the ROM's disk I/O routines to read more of SOS into memory.

The disk /// driver that is built into SOS also uses the ROM to perform disk block I/O (reference: DISK3.SRC). It is interesting to note that when the disk driver is initialized the driver checks if the ROM revision is 0 or 1. A revision of 0 is detected if address F1B9 contains 60. If neither revision is found then the disk driver returns an error to SOS (I don't think this will ever happen since the SOS loader has already determined that the ROM is revision 1). For a valid ROM revision the disk driver sets up several jump vectors which point to the appropriate addresses in the ROM for the various ROM routines needed by the disk driver. Therefore, the disk driver seems compatible with either ROM revision whereas the SOS loader likes only revision 1.

The .CONSOLE driver source listing appears to not use any ROM routines even though the ROM contains 40 and 80 column text routines and keyboard input routines. I assume the console driver was much more sophisticated than the ROM's text features and so using the ROM routines would not have worked well for this driver. I also assume that if the console driver used the ROM that when ROM

revision 1 was built the console driver would have had to be changed and Apple (smartly) did not want to do this.

6 MONITOR COMMANDS

Holding down the Open Apple and Control keys when the /// starts or when you press the Reset key activates the /// ROM Monitor. The screen will display in the upper left corner a small right-facing arrow with a blinking underscore character as the cursor. The Monitor's commands are based on the Apple][‘s Monitor commands but some commands have changed slightly and others are new for the (newer) ///.

The Monitor supports the following commands:

addr1.addr2	Dump memory data to screen from address 1 to address 2 and display ASCII character at the right of the screen.
CARRIAGE RETURN	Dump next line of addresses to the screen.
SPACE	Pause current memory dump. Press again to continue.
addr:byte_list	Store starting at the address the list of bytes.
addr:'text'	Store text starting at address with high bit clear.
addr:"text"	Store text starting at address with high bit set.
addr3<addr1.addr2M	Move data in addresses 1-2 to address 3.
addr3<addr1.addr2V	Verify data in addresses 1-2 is the same as data starting at address 3.
byte<addr1.addr2S	Search memory in address range 1-2 for the byte.
block<addr1.addr2W	Write address range to disk starting at the disk block.
block<addr1.addr2R	Read disk starting at block to the address range.
addrG	Call subroutine at the address.
addrJ	Jump to the address.
U	Call user routine starting at address $03F8.
X	Repeat last command line until you press the SPACE BAR.
ESC-8	Display 80 columns of text.
ESC-4	Display 40 columns of text.
/	Seperate multiple commands on the same line.
CTRL-I	Interrupt current operation, return to Monitor command line.

Note: See Wells' *Apple /// Entry Points* article for a great overview of the ROM Monitor, its commands (with some syntax errors), and the memory locations that need setting up for the key ROM routines to work. Apple's */// Service Reference Manual* (p. 13.57) has a list of Monitor commands. Anderson's *The Apple Nobody Knows* also has good Monitor command info.

To obtain a binary dump of the /// ROM you can do the following:

1. Initialize a disk on either the /// or an Apple][computer.
2. Insert the new disk in the ///.
3. Start the /// and hold down the Open Apple and Control keys.
4. You should be in the /// Monitor.
5. Type 0<F000.FFFW to write the ROM to disk blocks 0 to 7
6. Use a disk block reader on the /// or the][to read the ROM blocks and save them to a real file.

This disk writing is needed since the ROM does not provide a command for redirecting screen output to the ///'s serial port. But, I've read that you can output the ROM contents to the ///'s serial port but this involves using the Monitor to write a small program. If anyone has such a program please send a copy my way.

7 A FEW COMMENTS

I find it interesting, at least from a software engineering perspective, to note that in my opinion the /// ROM is missing several key features which I thought any system ROM would need. The ROM is missing two features which I think would have been useful to Apple and outside /// programmers:

1) The ROM does not have an explicit version number which exists at a specific ROM address. This version number could be used to validate the ROM in case there were several different ROMs (as there were). Apple uses a pseudo ROM version number (called the revision number) during the loading of SOS but this is somewhat lame in my opinion.

2) The ROM does not have a dispatch routine for use by the OS or applications that want to use ROM routines. This dispatch routine would reside at a specific address (e.g., F000) and it would take as input a command number and a set of parameters. These parameters could be passed via registers or on the stack. This routine would allow Apple to change the ROM and ROM "users" would not need to change their programming as long as they used the selector routine. The Apple][ROM did not have such a routine which caused Apple many headaches when it wanted to change the Apple][ROM and had to keep lots of routines in their same place.

3) The ROM source code is rather sparse concerning comments. It would be nice if the ROM contained detailed information about what each routine did and how to call the routines. Obviously, Apple did not expect anyone but Apple's own programmers to ever see the ROM source or use the ROM routines. (I've seen the Lisa computer's ROM listing which is much better documented than the ///'s and both are comparable in terms of age).

8 REFERENCES

Apple /// ROM Listing - Revision 0

This can be found in the Apple /// patent (#4,383,296) dated May 1983. Note that in places this ROM listing is not always readable.

Apple /// ROM Listing - Revision 1

I have a very readable listing of the revision 1 ROM that was printed on a laser printer.

Apple /// Service Reference Manual (Level 2)

This almost 500 page document by Apple has everything you would want to know about the ///'s hardware, low-level software, and how to service a broken ///. Includes descriptions of the System Monitor (a.k.a. Development Monitor) [page 17.3] and the built-in RAM test routine [page 13.51].

Apple /// SOS Bootstrap Loader Listing

Shows how 512 bytes of code are used to load SOS from disk into the ///'s memory.

The following articles provide good ROM information:

Apple /// Entry Points, Andy Wells, Call-APPLE, October 1981

Apple /// Dabbling, Rick Smith, Apple Orchard, Summer 1981

/// Bits: John Jeppson's Guided Tour of Highway ///, John Jeppson, Softalk, May 1983

The Apple Nobody Knows, Alan Anderson, Apple Orchard, Fall 1981

Unlocking the Apple /// - Part 3, Alan Anderson, Apple Orchard, September 1982

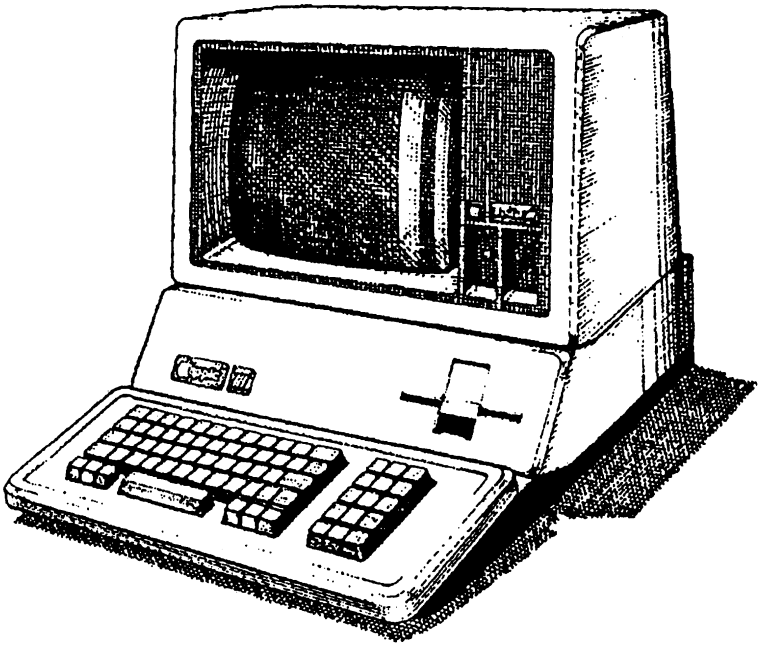
Apple ///: 12-Volt 128K Internal Diagnostics, Apple Technical Information Library

9 DOCUMENT MODIFICATION HISTORY

30 Nov 1997	Created this document.
04 Dec 1997	Corrected a few problems, extended the Reference section to include more /// articles pertaining to the /// ROM, added this section, added section MONITOR COMMANDS.

###

Apple III Computer Information



Inside the Apple III ROM

Revision 1 • 30 Nov 1997

"_183.PICT" 327 KB 2001-08-13 dpi: 600h x 600v pix: 4297h x 5030v

Inside the Apple /// Computer ROM

David T. Craig • 30 November 1997
71533.606@compuserve.com

TABLE OF CONTENTS

1 INTRODUCTION

This document provides a general overview of the contents of the Apple /// computer ROM revision 1. This information should be used in conjunction with a copy of the ROM source code listing. The audience of this document is anyone with an interest in the technology of the Apple /// computer's hardware and software.

NOTE

There were two revisions of the Apple /// ROM, revision 0 and revision 1. Revision 0 ROMs had at address F1B9 the value 60. Revision 1 ROMs had at address F1B9 the value A0.

This ROM contains 4 KB of 6502 programming and several data tables. The ROM occupies memory addresses F000-FFFF. The basic purpose of the ROM is to test the Apple /// computer hardware and boot an operating system from the ///'s built-in floppy disk drive. The ROM also contains a simple Monitor program whose purpose is to allow the user to interact with the /// at the hexadecimal level.

When the Apple /// computer is turned on the ROM's flow of execution is as follows:

1) The ROM starts execution at the address contained in FFFC-FFFD (RESET) which is address F4EE (DIAGN).

2) Diagnostics (DIAGN/F4EE) starts. The diagnostic first initializes some memory for the ROM's use. If the Open Apple key is held down then enter the ROM Monitor. Otherwise run several diagnostic checks of the /// hardware (tests zero page, sizes memory, initializes screen text buffer,

tests stack memory, tests ROM checksum, tests VIA chip, tests ACIA chip, tests A/D circuitry, tests keyboard connection). Any diagnostic failures display an error message and the user has to reset the computer.

3) Read block 0 (512 bytes) to address A000 from the floppy disk in the built-in disk drive (BOOT/F6A1). If no disk is found or block 0 cannot be read then display "RETRY" and wait for the user to reset the computer. If the block is successfully read then execute the block contents (this is called the SOS Bootstrap Loader: see section ROM USAGE BY SOS).

2 ROM SECTIONS

Section	Address	Purpose
Disk I/O	F000-F4C4	Read and write floppy disk blocks (512 bytes each)
Diagnostics	F4C5-F7FE	Diagnose the /// hardware
Monitor	F7FF-FFFF	Interacts with user so user can do simple things

3 IMPORTANT ROM ROUTINES

BLOCKIO / F479	Reads or write a disk block (512 bytes), calls routine REGRWTS (F000) which reads a sector (256 bytes) from the disk
BOOT / F6A1	Read floppy disk block #0 into address A000, execute the block
ENTRY / F901	Monitor entry point
DIAGN / F4EE	Diagnostic entry point
USRENTRY/F6E6	Tests RAM and displays a table showing chip failures (users may execute this routine from the Monitor)

4 ROM TABLES

Here's a list of the important data tables in the ROM. This list does not include disk I/O tables.

Table Name / Address	Contents
CHRSET / FEC5-FFB3	Default character set (overridden when SOS loads the character set from SOS.DRIVER)
Copyright / FFC0-FFEF	Copyright message (contains the initials "JRH" for J. R. Huston who was a key player behind the /// and SOS)
NMI / FFFA-FFFB	Jump address for the Non-Maskable Interrupt signal
RESET / FFFC-FFFD	Jump address when the /// is powered on

IRQ / FFFE-FFFF Jump address for the Interrupt Request signal

5 ROM USAGE BY SOS

The Apple /// operating system (SOS) uses several ROM routines. These routines seem to all be related to disk block I/O. The following discussion is based on SOS version 1.3.

When the ROM loads block 0 from a SOS disk the ROM is loading the SOS Bootstrap Loader program. This program, which is at most 512 bytes in length, uses the ROM routine REGRWTS (F000) to read the SOS Loader into memory. This program does not test the ROM revision. It is interesting to note that ROM routine BLOCKIO is not used, instead a lower-level routine (REGRWTS) is used.

The SOS Loader determines if the ROM is revision 1 by comparing address F1B9's contents against A0 (reference: SOS source file SOSLDR.D.SRC). If this comparison fails then SOS displays on the screen the error "ROM ERROR: PLEASE NOTIFY YOUR DEALER." If the ROM revision is correct then the SOS loader uses the ROM's disk I/O routines to read more of SOS into memory.

The disk /// driver that is built into SOS also uses the ROM to perform disk block I/O (reference: DISK3.SRC). It is interesting to note that when the disk driver is initialized the driver checks if the ROM revision is 0 or 1. A revision of 0 is detected if address F1B9 contains 60. If neither revision is found then the disk driver returns an error to SOS (I don't think this will ever happen since the SOS loader has already determined that the ROM is revision 1). For a valid ROM revision the disk driver sets up several jump vectors which point to the appropriate addresses in the ROM for the various ROM routines needed by the disk driver. Therefore, the disk driver seems compatible with either ROM revision whereas the SOS loader likes only revision 1.

6 A FEW COMMENTS

I find it interesting, at least from a software engineering perspective, that the ROM is missing some key features which I thought any system ROM would need. The ROM is missing two features which I think would have been useful to Apple and outside /// programmers:

1) The ROM does not have an explicit version number which exists at a specific ROM address. This version number could be used to validate the ROM in case there were several different ROMs (as there were). Apple uses a pseudo ROM version number (called the revision number) during the loading of SOS but this is somewhat lame in my opinion.

2) The ROM does not have a selector routine for use by the OS or applications that want to use ROM routines. This selector would reside at a specific address (e.g., F000) and it would take as input a command number and a set of parameters. These parameters could be passed via registers or on the stack. This routine would allow Apple to change the ROM and ROM

"users" would not need to change their programming as long as they used the selector routine. The Apple][ROM did not have such a routine which caused Apple many headaches when it wanted to change the Apple][ROM and had to keep lots of routines in their same place.

7 REFERENCES

Apple /// ROM Listing

I have a very nice listing of revision 1 ROM. A listing (that is somewhat readable) for the earlier revision 0 ROM may be found in the Apple /// patent.

Apple /// Service Reference Manual (Level 2)

This almost 500 page book by Apple has everything you would want to know about the ///'s hardware, low-level software, and how to service a broken ///. Includes descriptions of the System Monitor (a.k.a. Development Monitor) [page 17.3] and the built-in RAM test routine [page 13.51].

Apple /// SOS Bootstrap Loader Listing

Shows how 512 bytes of code is used to load SOS from disk into the ///'s memory.

###