

**Apple III**

SOS

Reference Manual, Volume 1



# Contents

## Volume 1: How SOS Works

### **Figures and Tables** xi

### **Preface** xvii

xvii	Scope Of This Manual
xviii	Using this Manual
xviii	About the Examples
xviii	Notation and Symbols
xviii	Numeric Notation
xix	Special Symbols

### **1 The Abstract Machine** 1

2	1.1 About Operating Systems
2	1.1.1 An Abstract Machine
2	1.1.2 A Resource Manager
3	1.1.3 A Common Foundation for Software
3	1.2 Overview of the Apple III
5	1.2.1 The Interpreter
5	1.2.2 SOS
6	1.2.3 Memory
7	1.2.4 Files
8	1.2.5 Devices
8	1.2.6 The 6502 Instruction Set

## **2** *Programs and Memory*

---

10	2.1	Addressing Modes
10	2.1.1	Bank-Switched Memory Addressing
13	2.1.2	Enhanced Indirect Addressing
16	2.2	Execution Environments
17	2.2.1	Zero Page and Stack
18	2.2.2	The Interpreter Environment
19	2.2.3	SOS Kernel Environment
20	2.2.4	SOS Device Driver Environment
22	2.2.5	Environment Summary
23	2.3	Segment Address Notation
25	2.3.1	Memory Calls
27	2.4	Memory Access Techniques
27	2.4.1	Subroutine and Module Addressing
29	2.4.2	Data Access
30	2.4.2.1	Bank-Switched Addressing
31	2.4.2.2	Enhanced Indirect Addressing
32	2.4.3	Address Conversion
33	2.4.3.1	Segment to Bank-Switched
33	2.4.3.2	Segment to Extended
34	2.4.3.3	Extended to Bank-Switched
36	2.4.4	Pointer Manipulation
36	2.4.4.1	Incrementing a Pointer
37	2.4.4.2	Comparing Two Pointers
38	2.4.5	Summary of Address Storage

## **3** *Devices*

---

39

- 40 3.1 Devices and Drivers
  - 40 3.1.1 Block and Character Devices
  - 40 3.1.2 Physical Devices and Logical Devices
  - 41 3.1.3 Device Drivers and Driver Modules
  - 41 3.1.4 Device Names
- 43 3.2 The SOS Device System
- 43 3.3 Device Information
- 45 3.4 Operations on Devices
- 46 3.5 Device Calls

## **4** *Files*

---

49

- 50 4.1 Character and Block Files
  - 50 4.1.1 Structure of Character and Block Files
  - 52 4.1.2 Open and Closed Files
  - 53 4.1.3 Volumes
    - 54 4.1.3.1 Volume Switching
    - 55 4.1.3.2 Volume Names
- 56 4.2 The SOS File System
  - 57 4.2.1 Directory Files and Standard Files
  - 58 4.2.2 File Names
  - 59 4.2.3 Pathnames
  - 61 4.2.4 The Prefix and Partial Pathnames
- 62 4.3 File and Access Path Information
  - 62 4.3.1 File Information
  - 64 4.3.2 Access Path Information
  - 67 4.3.3 Newline Mode Information
- 68 4.4 Operations on Files
- 69 4.5 File Calls

## 5 *File Organization on Block Devices*

---

75

- 77 5.1 Format of Information on a Volume (SOS 1.2)
- 78 5.2 Format of Directory Files
  - 79 5.2.1 Pointer Fields
  - 79 5.2.2 Volume Directory Headers
  - 82 5.2.3 Subdirectory Headers
  - 85 5.2.4 File Entries
  - 89 5.2.5 Field Formats in Detail
    - 89 5.2.5.1 The **storage\_type** Field
    - 89 5.2.5.2 The **creation** and **last\_mod** Fields
    - 90 5.2.5.3 The **access** Attributes
    - 91 5.2.5.4 The **file\_type** Field
  - 91 5.2.6 Reading a Directory File
- 92 5.3 Storage Formats of Standard Files
  - 92 5.3.1 Growing a Tree File
  - 95 5.3.2 Seedling Files
  - 95 5.3.3 Sapling Files
  - 96 5.3.4 Tree Files
  - 97 5.3.5 Sparse Files
  - 98 5.3.6 Locating a Byte in a Standard File
- 99 5.4 Chapter Overview

## 6 *Events and Resources*

---

103

- 104 6.1 Interrupts and Events
  - 108 6.1.1 Arming and Disarming Events
  - 108 6.1.2 The Event Queue
  - 109 6.1.3 The Event Fence
  - 110 6.1.4 Event Handlers
  - 112 6.1.5 Summary of Interrupts and Events
- 112 6.2 Resources
  - 112 6.2.1 The Clock
  - 113 6.2.2 The Analog Inputs
  - 114 6.2.3 TERMINATE
- 114 6.3 Utility Calls

## **7** *Interpreters and Modules*

---

117

- 118 7.1 Interpreters
  - 119 7.1.1 Structure of an Interpreter
  - 121 7.1.2 Obtaining Free Memory
  - 125 7.1.3 Event Arming and Response
- 125 7.2 A Sample Interpreter
  - 131 7.2.1 Complete Sample Listing
- 143 7.3 Creating Interpreter Files
- 143 7.4 Assembly-Language Modules
  - 144 7.4.1 Using Your Own Modules
  - 145 7.4.2 BASIC and Pascal Modules
  - 146 7.4.3 Creating Modules

## **8** *Making SOS Calls*

---

147

- 148 8.1 Types of SOS Calls
- 148 8.2 Form of a SOS Call
  - 148 8.2.1 The Call Block
  - 150 8.2.2 The Required Parameter List
  - 152 8.2.3 The Optional Parameter List
- 154 8.3 Pointer Address Extension
  - 155 8.3.1 Direct Pointers
    - 155 8.3.1.1 Direct Pointers to X-Bank Locations
    - 156 8.3.1.2 Direct Pointers to Current Bank Locations
  - 156 8.3.2 Indirect Pointers
    - 157 8.3.2.1 Indirect Pointers with an X-Byte of \$00
    - 158 8.3.2.2 Indirect Pointers with an X-Byte Between \$80 and \$8F
- 159 8.4 Name Parameters
- 160 8.5 SOS Call Error Reporting

## ***Index***

---

163

## **Volume 2: The SOS Calls**

### **Figures and Tables** vii

---

### **Preface** ix

---

## **9 File Calls and Errors** 1

---

- 2 9.1 File Calls
- 53 9.2 File Call Errors

## **10 Device Calls and Errors** 57

---

- 58 10.1 Device Calls
- 71 10.2 Device Call Errors

## **11 Memory Calls and Errors** 73

---

- 74 11.1 Memory Calls
- 88 11.2 Memory Call Errors

## **12 Utility Calls and Errors** 89

---

- 90 12.1 Utility Calls
- 104 12.2 Utility Call Errors

## **A** ***SOS Specifications*** **105**

---

- 106 Version
- 106 Classification
- 106 CPU Architecture
- 106 System Calls
- 106 File Management System
- 107 Device Management System
- 108 Memory/Buffer Management System
- 108 Additional System Functions
- 109 Interrupt Management System
- 109 Event Management System
- 109 System Configuration
- 109 Standard Device Drivers

## **B** ***ExerSOS*** **113**

---

- 114 B.1 Using ExerSOS
- 117 B.2 The Data Buffer
- 118 B.3 The String Buffer
- 119 B.4 Leaving ExerSOS

## **C** ***Make Interp*** **121**

---

## **D** ***Error Messages*** **123**

---

- 124 D.1 Non-Fatal SOS Errors
- 126 D.2 Fatal SOS Errors
- 128 D.3 Bootstrap Errors



## **E** ***Data Formats of Assembly-Language Code Files*** **131**

---

- 132 E.1 Code File Organization
- 134 E.2 The Segment Dictionary
- 135 E.3 The Code Part of a Code File

## ***Bibliography*** **141**

---

## ***Index*** **143**

---

## ***Figures and Tables***

### ***Volume 1: How SOS Works***

#### ***Preface***

---

xvii

xix Table 0-1 Numeric Notation

#### ***1 The Abstract Machine***

---

1

4 Figure 1-1 The Apple III/SOS Abstract Machine

#### ***2 Programs and Memory***

---

9

- 11 Figure 2-1 Bank-Switched Memory Addressing
- 12 Figure 2-2 Switching in Another Bank
- 14 Figure 2-3 X-byte Format
- 14 Figure 2-4 Enhanced Indirect Addressing
- 18 Figure 2-5 Interpreter Memory Placement
- 20 Figure 2-6 SOS Kernel Memory Placement
- 21 Figure 2-7 SOS Device Driver Memory Placement
- 23 Figure 2-8 Free Memory
- 24 Figure 2-9 Segment Address Notation
- 36 Figure 2-10 Increment Path

13	Table 2-1	Addresses in Bank-Switched Notation
16	Table 2-2	Extended Addresses
19	Table 2-3	Interpreter Environment
20	Table 2-4	SOS Kernel Environment
21	Table 2-5	SOS Device Driver Environment
22	Table 2-6	Environment Summary
24	Table 2-7	Addresses in Segment Notation
25	Table 2-8	Addresses in Segment Notation, S-Bank

### **3** **Devices** **39**

---

42	Figure 3-1	Device Name Syntax
43	Figure 3-2	The SOS Device System

### **4** **Files** **49**

---

51	Figure 4-1	Character File Model
51	Figure 4-2	Block File Model
52	Figure 4-3	Open Files
55	Figure 4-4	The SOS Disk Request
57	Figure 4-5	Top-Level Files
58	Figure 4-6	The SOS File System
59	Figure 4-7	File Name Syntax
60	Figure 4-8	Pathname Syntax
61	Figure 4-9	Pathnames
65	Figure 4-10	Automatic Movement of EOF and Mark
66	Figure 4-11	Manual Movement of EOF and Mark

## **5** *File Organization on Block Devices* 75

---

- 77 Figure 5-1 Blocks on a Volume
- 78 Figure 5-2 Directory File Format
- 80 Figure 5-3 The Volume Directory Header
- 83 Figure 5-4 The Subdirectory Header
- 86 Figure 5-5 The File Entry
- 90 Figure 5-6 Date and Time Format
- 90 Figure 5-7 The **access** Attribute Field
- 95 Figure 5-8 Structure of a Seedling File
- 96 Figure 5-9 Structure of a Sapling File
- 96 Figure 5-10 The Structure of a Tree File
- 98 Figure 5-11 A Sparse File
- 99 Figure 5-12 Format of **mark**
- 100 Figure 5-13 Disk Organization
- 102 Figure 5-14 Header and Entry Fields

## **6** *Events and Resources* 103

---

- 106 Figure 6-1 Queuing An Event
- 106 Figure 6-2 Handling An Event: Case A
- 107 Figure 6-3 Handling An Event: Case B
- 109 Figure 6-4 The Event Queue
- 110 Figure 6-5 The Event Fence
- 111 Figure 6-6 System Status during Event Handling

## **7** ***Interpreters and Modules*** **117**

---

- 119 Figure 7-1 Structure of an Interpreter
- 144 Figure 7-2 Interpreter and Modules

## **8** ***Making SOS Calls*** **147**

---

- 149 Figure 8-1 SOS Call Block
- 151 Figure 8-2 The Required Parameter List
- 153 Figure 8-3 Optional Parameter List
- 155 Figure 8-4 A Direct Pointer
- 157 Figure 8-5 An Indirect Pointer
- 159 Figure 8-6 Format of a Name Parameter

## **Volume 2: The SOS Calls**

### **Preface**

---

ix

- x Figure 0-1 Parts of the SOS Call
- xi Figure 0-2 TERMINATE Call Block

### **10 Device Calls and Errors**

---

57

- 60 Figure 10-1 Block Device Status Request \$00
- 60 Figure 10-2 Character Device Status Request \$01
- 61 Figure 10-3 Character Device Status Request \$02
- 64 Figure 10-4 Character Device Control Code \$01
- 64 Figure 10-5 Character Device Control Code \$02

### **E Data Formats of Assembly-Language Code Files**

---

131

- 133 Figure E-1 An Assembly-Language Code File
- 134 Figure E-2 A Segment Dictionary
- 135 Figure E-3 The Code Part of a Code File
- 137 Figure E-4 An Assembly-Language Procedure Attribute Table



## Preface

For your convenience and ease of reference, this manual is divided into two volumes. Volume 1: *How SOS Works* describes the operating system of the Apple III. Volume 2: *The SOS Calls* defines the individual SOS calls. Notice that the sequence of chapter numbers in Volume 1 continues unchanged into Volume 2.

### ***Scope of this Manual***

---

This manual describes SOS (pronounced “sauce”), the Sophisticated Operating System of the Apple III. With the information in this manual you’ll be able to write assembly-language programs that use the full power of the Apple III.

However, this manual is not a course in assembly-language programming. It assumes that you can program in assembly language and know the architecture of the 6502 microprocessor upon which the Apple III is based; it will explain how the architecture of the Apple III processor goes beyond that of the standard 6502. If you need more information on 6502 assembly-language programming, refer to one of the books listed in the bibliography of this manual.

The companion volume to this manual, the *Apple III SOS Device Driver Writer’s Guide*, contains the information you may need about the interface hardware of the Apple III, and tells how to create device drivers to use that hardware. If you wish to create custom interface software or hardware for the Apple III, read the present manual before turning to the *Apple III SOS Device Driver Writer’s Guide*.



## ***Using this Manual***

---

Before you begin with this manual, you should prepare yourself by reading the following:

- *the Apple III Owner's Guide* introduces you to some of the fundamental features of the Apple III—features that you will be exploring more deeply in this manual;
- *the Apple III Standard Device Drivers Manual* describes the workings of the Apple III's video screen, keyboard, graphics, and communications interfaces;
- *the Apple III Pascal Program Preparation Tools* manual explains the use of the Apple III Pascal Assembler, which is the only assembler that works with SOS.

You should also finish reading this preface, to learn about the notation and examples used in this manual.

## ***About the Examples***

---

Included in this manual are many sample programs and code fragments. These are intended as demonstrations *only*. In order to illustrate their concepts as well as possible, they are written to be clear and concise, without necessarily being efficient or comprehensive.

## ***Notation and Symbols***

---

Some special symbols and numeric notations are used throughout this manual.

### ***Numeric Notation***

We assume that you are familiar with the hexadecimal (hex) numbering system. All hexadecimal numbers in the text and tables of this manual are preceded by a dollar sign (\$). Any number in the text, a table, or illustration that is not preceded by a dollar sign is a decimal number.

Program listings from the Apple III Pascal Assembler, however, do not prefix hex numbers with dollar signs. In such listings, you can distinguish decimal numbers from hex by the fact that decimal numbers end with a decimal point (.). You can distinguish hex numbers from labels by the fact that hex numbers always begin with a digit from 0 to 9, and labels always begin with a letter.

Type	Notation in Text	Notation in Listings
Decimal	255	255.
Hexadecimal	\$3A5	3A5
Hexadecimal	\$BAD1	ØBAD1
Label	BAD1	BAD1

**Table 0-1.** Numeric Notation

Additional notations are introduced in Chapter 1.

## *Special Symbols*

Four special symbols are used in this manual to emphasize information about helpful or unusual features of the system.



This symbol precedes a paragraph that contains especially useful information.



Watch out! This symbol precedes a paragraph that warns you to be careful.



Stop! This symbol precedes a paragraph warning you that you are about to destroy data or harm hardware.



This symbol precedes a paragraph that is specific to versions 1.1, 1.2, and 1.3 of SOS. Note especially that, although the symbol indicates version 1.2, it is also applicable to versions 1.1 and 1.3.



# *The Abstract Machine*

- 2 1.1 About Operating Systems
  - 2 1.1.1 An Abstract Machine
  - 2 1.1.2 A Resource Manager
  - 3 1.1.3 A Common Foundation for Software
- 3 1.2 Overview of the Apple III
  - 5 1.2.1 The Interpreter
  - 5 1.2.2 SOS
  - 6 1.2.3 Memory
  - 7 1.2.4 Files
  - 8 1.2.5 Devices
  - 8 1.2.6 The 6502 Instruction Set

## 1.1 About Operating Systems

---

An *operating system* is the traffic controller of a computer system. A well-designed operating system increases the power and usefulness of a computer in three important ways. First, an operating system establishes an *abstract machine* that is defined by its concepts and models, rather than by the physical attributes of particular hardware. Second, it acts as a *resource manager*, to ease the programming task. Finally, it provides a *common foundation for software*.



If you are an experienced programmer of small computers, such as the Apple II, but you have never written large programs for a machine with an operating system, you should pay particular attention to this section.

### 1.1.1 An Abstract Machine

The low-level programming language of a computer is determined not only by its central processor, but by its operating system as well. The operating system is thus an essential part of the programming environment: knowing how it works lets you write programs that use the full power of the machine.

Most importantly, the combination of hardware and operating system software creates an abstract machine that is neither the hardware nor the operating system, but a synthesis of both. This is the machine you program.

The major advantage of the abstract-machine concept is that a program written for the abstract machine is not bound by the current configuration of the hardware. The operating system can compensate for expansions, enhancements, or changes in hardware, making these changes invisible to the programs. Thus programs properly written for an abstract machine need not be modified to respond to changes or improvements in the hardware.

### 1.1.2 A Resource Manager

An operating system also controls the flow of information into, out of, and within the computer. It provides standard ways to store and retrieve