Copyright © 1985 by Apple Computer, Inc.

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, mechanical, electronic, photocopying, recording, or otherwise, without prior written permission of Apple Computer, Inc. Printed in the United States of America.

© Apple Computer, Inc., 1985
20525 Mariani Avenue
Cupertino, CA 95014
(408) 996-1010

Apple, the Apple logo, LaserWriter, Lisa, Macintosh, the Macintosh logo, and MacWorks are registered trademarks of Apple Computer, Inc.

MacDraw, MacPaint, and MacWrite are registered trademarks of Claris Corporation.

Simultaneously published in the United States and Canada.

Written by Caroline Rose with Bradley Hacker, Robert Anders, Katie Withey, Mark Metzler, Steve Chernicoff, Chris Espinosa, Andy Averill, Brent Davis, and Brian Howard, assisted by Sandy Tompkins-Leffler and Louella Pizzuti. Special thanks to Cary Clark and Scott Knaster.

This book was produced using the Apple Macintosh computer and the LaserWriter printer.

ISBN 0-201-17733-1
KLMNOPQRSTUVWXYZ-MU-93210
11th printing, June 1990
WARRANTY INFORMATION

ALL IMPLIED WARRANTIES ON THIS MANUAL, INCLUDING IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, ARE LIMITED IN DURATION TO NINETY (90) DAYS FROM THE DATE OF THE ORIGINAL RETAIL PURCHASE OF THIS PRODUCT.

Even though Apple has reviewed this manual, APPLE MAKES NO WARRANTY OR REPRESENTATION, EITHER EXPRESS OR IMPLIED, WITH RESPECT TO THIS MANUAL, ITS QUALITY, ACCURACY, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. AS A RESULT, THIS MANUAL IS SOLD “AS IS,” AND YOU, THE PURCHASER, ARE ASSUMING THE ENTIRE RISK AS TO ITS QUALITY AND ACCURACY.

IN NO EVENT WILL APPLE BE LIABLE FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM ANY DEFECT OR INACCURACY IN THIS MANUAL, even if advised of the possibility of such damages.

THE WARRANTY AND REMEDIES SET FORTH ABOVE ARE EXCLUSIVE AND IN LIEU OF ALL OTHERS, ORAL OR WRITTEN, EXPRESS OR IMPLIED. No Apple dealer, agent, or employee is authorized to make any modification, extension, or addition to this warranty.

Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you. This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.
Contents

1 Preface
3 About Inside Macintosh
4 A Horse of a Different Color
5 The Structure of a Typical Chapter
5 Conventions

7 1 Finder Interface
9 About This Chapter
9 Signatures and File Types

15 2 The Macintosh Hardware
17 About This Chapter
17 Overview of the Hardware
18 The Video Interface
20 The Sound Generator
22 The SCC
25 The Mouse
29 The Keyboard and Keypad
33 The Disk Interface
36 The Real-Time Clock
39 The VIA
42 System Startup
43 Summary

47 3 Summary
49 About This Chapter
50 AppleTalk Manager
65 Binary-Decimal Conversion Package
66 Control Manager
71 Desk Manager
73 Device Manager
80 Dialog Manager
85 Disk Driver
88 Disk Initialization Package
90 Event Manager, Operating System
94 Event Manager, Toolbox
98 File Manager
113 Font Manager
118 International Utilities Package
Insider Macintosh

124 Memory Manager
130 Menu Manager
134 Package Manager
135 Printing Manager
141 QuickDraw
154 Resource Manager
157 Scrap Manager
159 Segment Loader
161 Serial Drivers
165 Sound Driver
172 Standard File Package
176 System Error Handler
179 TextEdit
184 Utilities, Operating System
190 Utilities, Toolbox
193 Vertical Retrace Manager
195 Window Manager
201 Assembly Language

205 Appendix A: Result Codes
211 Appendix B: Routines That May Move or Purge Memory
215 Appendix C: System Traps
227 Appendix D: Global Variables
233 Glossary
261 Index
PREFACE

3  About Inside Macintosh
3  The Language
4  What's in Each Volume
4  Version Numbers
4  A Horse of a Different Color
5  The Structure of a Typical Chapter
5  Conventions
ABOUT INSIDE MACINTOSH

*Inside Macintosh* is a three-volume set of manuals that tells you what you need to know to write software for the Apple® Macintosh™ 128K, 512K, or XL (or a Lisa® running MacWorks™ XL). Although directed mainly toward programmers writing standard Macintosh applications, *Inside Macintosh* also contains the information needed to write simple utility programs, desk accessories, device drivers, or any other Macintosh software. It includes:

- the user interface guidelines for applications on the Macintosh
- a complete description of the routines available for your program to call (both those built into the Macintosh and others on disk), along with related concepts and background information
- a description of the Macintosh 128K and 512K hardware

It does not include information about:

- Programming in general.
- Getting started as a developer. For this, write to:
  
  Developer Relations  
  Mail Stop 27-S  
  Apple Computer, Inc.  
  20525 Mariani Avenue  
  Cupertino, CA 95014

- Any specific development system, except where indicated. You'll need to have additional documentation for the development system you're using.

- The Standard Apple Numeric Environment (SANE), which your program can access to perform extended-precision floating-point arithmetic and transcendental functions. This environment is described in the *Apple Numerics Manual*.

You should already be familiar with the basic information that's in *Macintosh*, the owner's guide, and have some experience using a standard Macintosh application (such as MacWrite™).

**The Language**

The routines you'll need to call are written in assembly language, but (with a few exceptions) they're also accessible from high-level languages, such as Pascal on the Lisa Workshop development system. *Inside Macintosh* documents the Lisa Pascal interfaces to the routines and the symbolic names defined for assembly-language programmers using the Lisa Workshop; if you're using a different development system, its documentation should tell you how to apply the information presented here to that system.

*Inside Macintosh* is intended to serve the needs of both high-level language and assembly-language programmers. Every routine is shown in its Pascal form (if it has one), but assembly-language programmers are told how they can access the routines. Information of interest only to assembly-language programmers is isolated and labeled so that other programmers can conveniently skip it.
Familiarity with Lisa Pascal (or a similar high-level language) is recommended for all readers, since it's used for most examples. Lisa Pascal is described in the documentation for the Lisa Pascal Workshop.

What's in Each Volume

*Inside Macintosh* consists of three volumes. Volume I begins with the following information of general interest:

- a "road map" to the software and the rest of the documentation
- the user interface guidelines
- an introduction to memory management (the least you need to know, with a complete discussion following in Volume II)
- some general information for assembly-language programmers

It then describes the various parts of the User Interface Toolbox, the software in ROM that helps you implement the standard Macintosh user interface in your application. This is followed by descriptions of other, RAM-based software that's similar in function to the User Interface Toolbox. (The software overview in the Road Map chapter gives further details.)

Volume II describes the Operating System, the software in ROM that does basic tasks such as input and output, memory management, and interrupt handling. As in Volume I, some functionally similar RAM-based software is then described.

Volume III discusses your program's interface with the Finder and then describes the Macintosh 128K and 512K hardware. A comprehensive summary of all the software is provided, followed by some useful appendices and a glossary of all terms defined in *Inside Macintosh*.

Version Numbers

This edition of *Inside Macintosh* describes the following versions of the software:

- version 105 of the ROM in the Macintosh 128K or 512K
- version 112 of the ROM image installed by MacWorks in the Macintosh XL
- version 1.1 of the Lisa Pascal interfaces and the assembly-language definitions

Some of the RAM-based software is read from the file named System (usually kept in the System Folder). This manual describes the software in the System file whose creation date is May 2, 1984.

A HORSE OF A DIFFERENT COLOR

On an innovative system like the Macintosh, programs don't look quite the way they do on other systems. For example, instead of carrying out a sequence of steps in a predetermined order, your program is driven primarily by user actions (such as clicking and typing) whose order cannot be predicted.
You'll probably find that many of your preconceptions about how to write applications don’t apply here. Because of this, and because of the sheer volume of information in Inside Macintosh, it’s essential that you read the Road Map chapter. It will help you get oriented and figure out where to go next.

THE STRUCTURE OF A TYPICAL CHAPTER

Most chapters of Inside Macintosh have the same structure, as described below. Reading through this now will save you a lot of time and effort later on. It contains important hints on how to find what you’re looking for within this vast amount of technical documentation.

Every chapter begins with a very brief description of its subject and a list of what you should already know before reading that chapter. Then there’s a section called, for example, "About the Window Manager", which gives you more information about the subject, telling you what you can do with it in general, elaborating on related user interface guidelines, and introducing terminology that will be used in the chapter. This is followed by a series of sections describing important related concepts and background information; unless they’re noted to be for advanced programmers only, you’ll have to read them in order to understand how to use the routines described later.

Before the routine descriptions themselves, there’s a section called, for example, "Using the Window Manager". It introduces you to the routines, telling you how they fit into the general flow of an application program and, most important, giving you an idea of which ones you’ll need to use. Often you’ll need only a few routines out of many to do basic operations; by reading this section, you can save yourself the trouble of learning routines you’ll never use.

Then, for the details about the routines, read on to the next section. It gives the calling sequence for each routine and describes all the parameters, effects, side effects, and so on.

Following the routine descriptions, there may be some sections that won't be of interest to all readers. Usually these contain information about advanced techniques, or behind the scenes details for the curious.

For review and quick reference, each chapter ends with a summary of the subject matter, including the entire Pascal interface and a separate section for assembly-language programmers.

CONVENTIONS

The following notations are used in Inside Macintosh to draw your attention to particular items of information:

Note: A note that may be interesting or useful

Warning: A point you need to be cautious about

Assembly-language note: A note of interest to assembly-language programmers only
Routines marked with this notation are not part of the Macintosh ROM. Depending on how the interfaces have been set up on the development system you’re using, these routines may or may not be available. They’re available to users of Lisa Pascal; other users should check the documentation for their development system for more information. (For related information of interest to assembly-language programmers, see chapter 4 of Volume I.)
1 THE FINDER INTERFACE

9 About This Chapter
9 Signatures and File Types
10Finder-Related Resources
10 Version Data
10Icons and File References
11Bundles
11An Example
12Formats of Finder-Related Resources
ABOUT THIS CHAPTER

This chapter describes the interface between a Macintosh application program and the Finder. You should already be familiar with the details of the User Interface Toolbox and the Operating System.

SIGNATURES AND FILE TYPES

Every application must have a unique signature by which the Finder can identify it. The signature can be any four-character sequence not being used for another application on any currently mounted volume (except that it can't be one of the standard resource types). To ensure uniqueness on all volumes, you must register your application's signature by writing to:

Macintosh Technical Support
Mail Stop 3-T
Apple Computer, Inc.
20525 Mariani Avenue
Cupertino, CA 95014

Note: There's no need to register your own resource types, since they'll usually exist only in your own applications or documents.

Signatures work together with file types to enable the user to open or print a document (any file created by an application) from the Finder. When the application creates a file, it sets the file's creator and file type. Normally it sets the creator to its signature and the file type to a four-character sequence that identifies files of that type. When the user asks the Finder to open or print the file, the Finder starts up the application whose signature is the file's creator and passes the file type to the application along with other identifying information, such as the file name. (More information about this process is given in chapter 2 of Volume II.)

An application may create its own special type or types of files. Like signatures, file types must be registered with Macintosh Technical Support to ensure uniqueness. When the user chooses Open from an application's File menu, the application will display (via the Standard File Package) the names of all files of a given type or types, regardless of which application created the files. Having a unique file type for your application's special files ensures that only the names of those files will be displayed for opening.

Note: Signatures and file types may be strange, unreadable combinations of characters; they're never seen by end users of Macintosh.

Applications may also create existing types of files. There might, for example, be an application that merges two MacWrite documents into a single document. In such cases, the application should use the same file type as the original application uses for those files. It should also specify the original application's signature as the file's creator; that way, when the user asks the Finder to open or print the file, the Finder will call on the original application to perform the operation. To learn the signature and file types used by an existing application, check with the application's manufacturer.
Files that consist only of text—a stream of characters, with Return characters at the ends of paragraphs or short lines—should be given the standard file type 'TEXT'. This is the type that MacWrite gives to text-only files it creates, for example. If your application uses this file type, its files will be accepted by MacWrite and it in turn will accept MacWrite text-only files (likewise for any other application that deals with 'TEXT' files, such as MacTerminal). Your application can give its own signature as the file's creator if it wants to be called to open or print the file when the user requests this from the Finder.

For files that aren't to be opened or printed from the Finder, as may be the case for certain data files created by the application, the creator should be set to '????' (and the file type to whatever is appropriate).

**FINDER-RELATED RESOURCES**

To establish the proper interface with the Finder, every application's resource file must specify the signature of the application along with data that provides version information. In addition, there may be resources that provide information about icons and files related to the application. All of these Finder-related resources are described below, followed by a comprehensive example and (for interested programmers) the exact formats of the resources.

**Version Data**

Your application's resource file must contain a special resource that has the signature of the application as its resource type. This resource is called the version data of the application. The version data is typically a string that gives the name, version number, and date of the application, but it can in fact be any data at all. The resource ID of the version data is 0 by convention.

Part of the process of installing an application on the Macintosh is to set the creator of the file that contains the application. You set the creator to the application's signature, and the Finder copies the corresponding version data into a resource file named Desktop. (The Finder doesn't display this file on the Macintosh desktop, to ensure that the user won't tamper with it.)

*Note:* Additional, related resources may be copied into the Desktop file; see "Bundles" below for more information.

**Icons and File References**

For each application, the Finder needs to know:

- the icon to be displayed for the application on the desktop, if different from the Finder's default icon for applications (see Figure 1)
- if the application creates any files, the icon to be displayed for each type of file it creates, if different from the Finder's default icon for documents

The Finder learns this information from resources called file references in the application's resource file. Each file reference contains a file type and an ID number, called a local ID, that identifies the icon to be displayed for that type of file. (The local ID is mapped to an actual resource ID as described under "Bundles" below.)

*III-10 Signatures and File Types*
The Finder Interface

Application  Document

Figure 1. The Finder's Default Icons

The file type for the application itself is 'APPL'. This is the file type in the file reference that designates the application's icon. You also specify it as the application's file type at the same time that you specify its creator—when you install the application on the Macintosh.

The ID number in a file reference corresponds not to a single icon but to an icon list in the application's resource file. The icon list consists of two icons: the actual icon to be displayed on the desktop, and a mask consisting of that icon's outline filled with black (see Figure 2).

Figure 2. Icon and Mask

Bundles

A bundle in the application's resource file groups together all the Finder-related resources. It specifies the following:

- the application's signature and the resource ID of its version data
- a mapping between the local IDs for icon lists (as specified in file references) and the actual resource IDs of the icon lists in the resource file
- local IDs for the file references themselves and a mapping to their actual resource IDs

When you install the application on the Macintosh, you set its "bundle bit"; the first time the Finder sees this, it copies the version data, bundle, icon lists, and file references from the application's resource file into the Desktop file. If there are any resource ID conflicts between the icon lists and file references in the application's resource file and those in Desktop, the Finder will change those resource IDs in Desktop. The Finder does this same resource copying and ID conflict resolution when you transfer an application to another volume.

Note: The local IDs are needed only for use by the Finder.

An Example

Suppose you've written an application named SampWriter. The user can create a unique type of document from it, and you want a distinctive icon for both the application and its documents. The application's signature, as recorded with Macintosh Technical Support, is 'SAMP'; the file type assigned for its documents is 'SAMF'. You would include the following resources in the application's resource file:
<table>
<thead>
<tr>
<th>Resource</th>
<th>Resource ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version data with resource type 'SAMP'</td>
<td>0</td>
<td>The string 'SampWriter Version 1--2/1/85'</td>
</tr>
<tr>
<td>Icon list</td>
<td>128</td>
<td>The icon for the application</td>
</tr>
<tr>
<td>Icon list</td>
<td>129</td>
<td>The icon's mask</td>
</tr>
<tr>
<td>Icon list</td>
<td>130</td>
<td>File type 'APPL'</td>
</tr>
<tr>
<td>Icon list</td>
<td>131</td>
<td>File type 'SAMF'</td>
</tr>
<tr>
<td>Bundle</td>
<td>132</td>
<td>Signature 'SAMP'</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Description</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>File reference</td>
<td>130</td>
</tr>
<tr>
<td>File reference</td>
<td>131</td>
</tr>
<tr>
<td>File type 'APPL'</td>
<td></td>
</tr>
<tr>
<td>Local ID 0 for the icon list</td>
<td></td>
</tr>
<tr>
<td>Local ID 1 for the icon list</td>
<td></td>
</tr>
<tr>
<td>Signature 'SAMP'</td>
<td></td>
</tr>
</tbody>
</table>

Note: See the documentation for the development system you're using for information about how to include these resources in a resource file.

**Formats of Finder-Related Resources**

The resource type for an application's version data is the signature of the application, and the resource ID is 0 by convention. The resource data can be anything at all; typically it's a string giving the name, version number, and date of the application.

The resource type for an icon list is 'ICN#'. The resource data simply consists of the icons, 128 bytes each.

The resource type for a file reference is 'FREF'. The resource data has the format shown below.

<table>
<thead>
<tr>
<th>Number of bytes</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 bytes</td>
<td>File type</td>
</tr>
<tr>
<td>2 bytes</td>
<td>Local ID for icon list</td>
</tr>
</tbody>
</table>

The resource type for a bundle is 'BNDL'. The resource data has the format shown below. The format is more general than needed for Finder-related purposes because bundles will be used in other ways in the future.
<table>
<thead>
<tr>
<th>Number of bytes</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 bytes</td>
<td>Signature of the application</td>
</tr>
<tr>
<td>2 bytes</td>
<td>Resource ID of version data</td>
</tr>
<tr>
<td>2 bytes</td>
<td>Number of resource types in bundle minus 1</td>
</tr>
</tbody>
</table>

For each resource type:

<table>
<thead>
<tr>
<th>4 bytes</th>
<th>Resource type</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 bytes</td>
<td>Number of resources of this type minus 1</td>
</tr>
</tbody>
</table>

For each resource:

<table>
<thead>
<tr>
<th>2 bytes</th>
<th>Local ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 bytes</td>
<td>Actual resource ID</td>
</tr>
</tbody>
</table>

A bundle used for establishing the Finder interface contains the two resource types 'ICN#' and 'FREF'.

*Finder-Related Resources III-13*
2 THE MACINTOSH HARDWARE

17 About This Chapter
17 Overview of the Hardware
18 The Video Interface
20 The Sound Generator
22 Diagram
22 The SCC
25 Diagram
25 The Mouse
28 Diagram
29 The Keyboard and Keypad
30 Keyboard Communication Protocol
31 Keypad Communication Protocol
33 The Disk Interface
34 Controlling the Disk-State Control Lines
34 Reading from the Disk Registers
35 Writing to the Disk Registers
36 Explanations of the Disk Registers
36 The Real-Time Clock
37 Accessing the Clock Chip
38 The One-Second Interrupt
39 The VIA
39 VIA Register A
39 VIA Register B
40 The VIA Peripheral Control Register
40 The VIA Timers
41 VIA Interrupts
42 Other VIA Registers
42 System Startup
43 Summary
ABOUT THIS CHAPTER

This chapter provides a basic description of the hardware of the Macintosh 128K and 512K computers. It gives you information that you'll need to connect other devices to the Macintosh and to write device drivers or other low-level programs. It will help you figure out which technical documents you'll need to design peripherals; in some cases, you'll have to obtain detailed specifications from the manufacturers of the various interface chips.

This chapter is oriented toward assembly-language programmers. It assumes you're familiar with the basic operation of microprocessor-based devices. Knowledge of the Macintosh Operating System will also be helpful.

Warning: Only the Macintosh 128K and 512K are covered in this chapter. In particular, note that the memory addresses and screen size are different on the Macintosh XL (and may be different in future versions of the Macintosh). To maintain software compatibility across the Macintosh line, and to allow for future changes to the hardware, you're strongly advised to use the Toolbox and Operating System routines wherever possible.

To learn how your program can determine which hardware environment it's operating in, see the description of the Environs procedure in chapter 13 of Volume II.

OVERVIEW OF THE HARDWARE

The Macintosh computer contains a Motorola MC68000 microprocessor clocked at 7.8336 megahertz, random access memory (RAM), read-only memory (ROM), and several chips that enable it to communicate with external devices. There are five I/O devices: the video display; the sound generator; a Synertek SY6522 Versatile Interface Adapter (VIA) for the mouse and keyboard; a Zilog Z8530 Serial Communications Controller (SCC) for serial communication; and an Apple custom chip, called the IWM ("Integrated Woz Machine") for disk control.

The Macintosh uses memory-mapped I/O, which means that each device in the system is accessed by reading or writing to specific locations in the address space of the computer. Each device contains logic that recognizes when it's being accessed and responds in the appropriate manner. The MC68000 can directly access 16 megabytes (Mb) of address space. In the Macintosh, this is divided into four equal sections. The first four Mb are for RAM, the second four Mb are for ROM, the third are for the SCC, and the last four are for the IWM and the VIA. Since each of the devices within the blocks has far fewer than four Mb of individually addressable locations or registers, the addresses within each block "wrap around" and are repeated several times within the block.

RAM is the "working memory" of the system. Its base address is address 0. The first 256 bytes of RAM (addresses 0 through $FF) are used by the MC68000 as exception vectors; these are the addresses of the routines that gain control whenever an exception such as an interrupt or a trap occurs. (The summary at the end of this chapter includes a list of all the exception vectors.) RAM also contains the system and application heaps, the stack, and other information used by applications. In addition, the following hardware devices share the use of RAM with the MC68000:
Inside Macintosh

- the video display, which reads the information for the display from one of two screen buffers
- the sound generator, which reads its information from one of two sound buffers
- the disk speed controller, which shares its data space with the sound buffers

The MC68000 accesses to RAM are interleaved (alternated) with the video display's accesses during the active portion of a screen scan line (video scanning is described in the next section). The sound generator and disk speed controller are given the first access after each scan line. At all other times, the MC68000 has uninterrupted access to RAM, increasing the average RAM access rate to about 6 megahertz (MHz).

ROM is the system's permanent read-only memory. Its base address, $400000, is available as the constant romStart and is also stored in the global variable ROMBase. ROM contains the routines for the Toolbox and Operating System, and the various system traps. Since the ROM is used exclusively by the MC68000, it's always accessed at the full processor rate of 7.83 MHz.

The address space reserved for the device I/O contains blocks devoted to each of the devices within the computer. This region begins at address $800000 and continues to the highest address at $FFFFFF.

Note: Since the VIA is involved in some way in almost every operation of the Macintosh, the following sections frequently refer to the VIA and VIA-related constants. The VIA itself is described later, and all the constants are listed in the summary at the end of this chapter.

THE VIDEO INTERFACE

The video display is created by a moving electron beam that scans across the screen, turning on and off as it scans in order to create black and white pixels. Each pixel is a square, approximately 1/74 inch on a side.

To create a screen image, the electron beam starts at the top left corner of the screen (see Figure 1). The beam scans horizontally across the screen from left to right, creating the top line of graphics. When it reaches the last pixel on the right end of the top line it turns off, and continues past the last pixel to the physical right edge of the screen. Then it flicks invisibly back to the left edge and moves down one scan line. After tracing across the black border, it begins displaying the data in the second scan line. The time between the display of the rightmost pixel on one line and the leftmost pixel on the next is called the horizontal blanking interval. When the electron beam reaches the last pixel of the last (342nd) line on the screen, it traces out to the right edge and then flicks up to the top left corner, where it traces the left border and then begins once again to display the top line. The time between the last pixel on the bottom line and the first one on the top line is called the vertical blanking interval. At the beginning of the vertical blanking interval, the VIA generates a vertical blanking interrupt.

The pixel clock rate (the frequency at which pixels are displayed) is 15.6672 MHz, or about .064 microseconds (μsec) per pixel. For each scan line, 512 pixels are drawn on the screen, requiring 32.68 μsec. The horizontal blanking interval takes the time of an additional 192 pixels, or 12.25 μsec. Thus, each full scan line takes 44.93 μsec, which means the horizontal scan rate is 22.25 kilohertz.
A full screen display consists of 342 horizontal scan lines, occupying 15367.65 µsec, or about 15.37 milliseconds (msec). The vertical blanking interval takes the time of an additional 28 scan lines—1258.17 µsec, or about 1.26 msec. This means the full screen is redisplayed once every 16625.8 µsec. That’s about 16.6 msec per frame, which means the vertical scan rate (the full screen display frequency) is 60.15 hertz.

The video generator uses 21,888 bytes of RAM to compose a bit-mapped video image 512 pixels wide by 342 pixels tall. Each bit in this range controls a single pixel in the image: A 0 bit is white, and a 1 bit is black.

There are two screen buffers (areas of memory from which the video circuitry can read information to create a screen display): the main buffer and the alternate buffer. The starting addresses of the screen buffers depend on how much memory you have in your Macintosh. In a Macintosh 128K, the main screen buffer starts at $1A700 and the alternate buffer starts at $12700; for a 512K Macintosh, add $60000 to these numbers.

**Warning:** To be sure you don't use the wrong area of memory and to maintain compatibility with future Macintosh systems, you should get the video base address and bit map dimensions from screenBits (see chapter 6 of Volume I).

Each scan line of the screen displays the contents of 32 consecutive words of memory, each word controlling 16 horizontally adjacent pixels. In each word, the high-order bit (bit 15) controls the leftmost pixel and the low-order bit (bit 0) controls the rightmost pixel. The first word in each scan line follows the last word on the line above it. The starting address of the screen is thus in
the top left corner, and the addresses progress from there to the right and down, to the last byte in
the extreme bottom right corner.

Normally, the video display doesn't flicker when you read from or write to it, because the video
memory accesses are interleaved with the processor accesses. But if you're creating an animated
image by repeatedly drawing the graphics in quick succession, it may appear to flicker if the
electron beam displays it when your program hasn't finished updating it, showing some of the
new image and some of the old in the same frame.

One way to prevent flickering when you're updating the screen continuously is to use the vertical
and horizontal blanking signals to synchronize your updates to the scanning of video memory.
Small changes to your screen can be completed entirely during the interval between frames (the
first 1.26 msec following a vertical blanking interrupt), when nothing is being displayed on the
screen. When making larger changes, the trick is to keep your changes happening always ahead
of the spot being displayed by the electron beam, as it scans byte by byte through the video
memory. Changes you make in the memory already passed over by the scan spot won't appear
until the next frame. If you start changing your image when the vertical blanking interrupt occurs,
you have 1.26 msec of unrestricted access to the image. After that, you can change progressively
less and less of your image as it's scanned onto the screen, starting from the top (the lowest video
memory address). From vertical blanking interrupt, you have only 1.26 msec in which to change
the first (lowest address) screen location, but you have almost 16.6 msec to change the last
(highest address) screen location.

Another way to create smooth, flicker-free graphics, especially useful with changes that may take
more 16.6 msec, is to use the two screen buffers as alternate displays. If you draw into the one
that's currently not being displayed, and then switch the buffers during the next vertical blanking,
your graphics will change all at once, producing a clean animation. (See chapter 11 of Volume II
to find out how to specify tasks to be performed during vertical blanking.)

If you want to use the alternate screen buffer, you'll have to specify this to the Segment Loader
(see chapter 2 of Volume II for details). To switch to the alternate screen buffer, clear the
following bit of VIA data register A (vBase+vBufA):

    vPage2 .EQU 6 ; 0 = alternate screen buffer

For example:

    BCLR #vPage2,vBase+vBufA

To switch back to the main buffer, set the same bit.

**Warning:** Whenever you change a bit in a VIA data register, be sure to leave the other
bits in the register unchanged.

**Warning:** The alternate screen buffer may not be supported in future versions of the
Macintosh.

---

**THE SOUND GENERATOR**

The Macintosh sound circuitry uses a series of values taken from an area of RAM to create a
changing waveform in the output signal. This signal drives a small speaker inside the Macintosh

---

III-20 *The Video Interface*
and is connected to the external sound jack on the back of the computer. If a plug is inserted into the external sound jack, the internal speaker is disabled. The external sound line can drive a load of 600 or more ohms, such as the input of almost any audio amplifier, but not a directly connected external speaker.

The sound generator may be turned on or off by writing 1 (off) or 0 (on) to the following bit of VIA data register B (vBase+vBufB):

\[ \text{vSndEnb} \quad \text{EQU} \quad 7 \quad ; 0 = \text{sound enabled}, \; 1 = \text{disabled} \]

For example:

\begin{verbatim}
BSET #vSndEnb,vBase+vBufB ; turn off sound
\end{verbatim}

By storing a range of values in the sound buffer, you can create the corresponding waveform in the sound channel. The sound generator uses a form of pulse-width encoding to create sounds. The sound circuitry reads one word in the sound buffer during each horizontal blanking interval (including the "virtual" intervals during vertical blanking) and uses the high-order byte of the word to generate a pulse of electricity whose duration (width) is proportional to the value in the byte. Another circuit converts this pulse into a voltage that's attenuated (reduced) by a three-bit value from the VIA. This reduction corresponds to the current setting of the volume level. To set the volume directly, store a three-bit number in the low-order bits of VIA data register A (vBase+vBufA). You can use the following constant to isolate the bits involved:

\[ \text{vSound} \quad \text{EQU} \quad 7 \quad ; \text{sound volume bits} \]

Here's an example of how to set the sound level:

\begin{verbatim}
MOVE.B vBase+vBufA,DO ; get current value of register A
ANDI.B #255-vSound,DO ; clear the sound bits
ORI.B #3,DO ; set medium sound level
MOVE.B DO,vBase+vBufA ; put the data back
\end{verbatim}

After attenuation, the sound signal is passed to the audio output line.

The sound circuitry scans the sound buffer at a fixed rate of 370 words per video frame, repeating the full cycle 60.15 times per second. To create sounds with frequencies other than multiples of the basic scan rate, you must store phase-shifted patterns into the sound buffer between each scan. You can use the vertical and horizontal blanking signals (available in the VIA) to synchronize your sound buffer updates to the buffer scan. You may find that it's much easier to use the routines in the Sound Driver to do these functions.

**Warning:** The low-order byte of each word in the sound buffer is used to control the speed of the motor in the disk drive. Don't store any information there, or you'll interfere with the disk I/O.

There are two sound buffers, just as there are two screen buffers. The address of the main sound buffer is stored in the global variable SoundBase and is also available as the constant soundLow. The main sound buffer is at $1FD00 in a 128K Macintosh, and the alternate buffer is at $1A100; for a 512K Macintosh, add $60000 to these values. Each sound buffer contains 370 words of data. As when you want to use the alternate screen buffer, you'll have to specify to the Segment Loader that you want the alternate buffer (see chapter 2 of Volume II for details). To select the alternate sound buffer for output, clear the following bit of VIA data register A (vBase+vBufA):
Inside Macintosh

vSndPg2 .EQU 3 ; 0 = alternate sound buffer

To return to the main buffer, set the same bit.

Warning: Be sure to switch back to the main sound buffer before doing a disk access, or the disk won't work properly.

Warning: The alternate sound buffer may not be supported in future versions of the Macintosh.

There's another way to generate a simple, square-wave tone of any frequency, using almost no processor intervention. To do this, first load a constant value into all 370 sound buffer locations (use 00's for minimum volume, $FF's for maximum volume). Next, load a value into the VIA's timer 1 latches, and set the high-order two bits of the VIA's auxiliary control register (vBase+vACR) for "square wave output" from timer 1. The timer will then count down from the latched value at 1.2766 µsec/count, over and over, inverting the vSndEnb bit of VIA register B (vBase+vBufB) after each count down. This takes the constant voltage being generated from the sound buffer and turns it on and off, creating a square-wave sound whose period is

\[ 2 \times 1.2766 \mu\text{sec} \times \text{timer 1's latched value} \]

Note: You may want to disable timer 1 interrupts during this process (bit 6 in the VIA's interrupt enable register, which is at vBase+vIER).

To stop the square-wave sound, reset the high-order two bits of the auxiliary control register.

Note: See the SY6522 technical specifications for details of the VIA registers. See also "Sound Driver Hardware" in chapter 8 of Volume II.

Diagram

Figure 2 shows a block diagram for the sound port.

THE SCC

The two serial ports are controlled by a Zilog Z8530 Serial Communications Controller (SCC). The port known as SCC port A is the one with the modem icon on the back of the Macintosh. SCC port B is the one with the printer icon.

Macintosh serial ports conform to the EIA standard RS422, which differs from the more common RS232C standard. While RS232C modulates a signal with respect to a common ground ("single-ended" transmission), RS422 modulates two signals against each other ("differential" transmission). The RS232C receiver senses whether the received signal is sufficiently negative with respect to ground to be a logic "1", whereas the RS422 receiver simply senses which line is more negative than the other. This makes RS422 more immune to noise and interference, and more versatile over longer distances. If you ground the positive side of each RS422 receiver and leave unconnected the positive side of each transmitter, you've converted to EIA standard RS423, which can be used to communicate with most RS232C devices over distances up to fifty feet or so.

III-22 The Sound Generator
The Macintosh Hardware

The Macintosh Hardware

Internal speaker
(disconnected when sound output connector is used)

Sound output connector

Sound Amplifier

Volume control (eight levels)

On-off switch (square-wave generator)

Digital-to-analog convertor

Digital-to-analog convertor

To motor speed control lines for internal and external disk drives

A new word is read every screen scan line (44.93 usec)

Figure 2. Diagram of Sound Port

The SCC III-23
The serial inputs and outputs of the SCC are connected to the ports through differential line drivers (26LS30) and receivers (26LS32). The line drivers can be put in high-impedance mode between transmissions, to allow other devices to transmit over those lines. A driver is activated by lowering the SCC's Request To Send (RTS) output for that port. Port A and port B are identical except that port A (the modem port) has a higher interrupt priority, making it more suitable for high-speed communication.

Figure 3 shows the DB-9 pinout for the SCC output jacks.

![Pinout for SCC Output Jacks](image)

**Warning:** Do not draw more than 100 milliamps at +12 volts, and 200 milliamps at +5 volts from all connectors combined.

Each port's input-only handshake line (pin 7) is connected to the SCC's Clear To Send (CTS) input for that port, and is designed to accept an external device's Data Terminal Ready (DTR) handshake signal. This line is also connected to the SCC's external synchronous clock (TRxC) input for that port, so that an external device can perform high-speed synchronous data exchange. Note that you can't use the line for receiving DTR if you're using it to receive a high-speed data clock.

The handshake line is sensed by the Macintosh using the positive (noninverting) input of one of the standard RS422 receivers (26LS32 chip), with the negative input grounded. The positive input was chosen because this configuration is more immune to noise when no active device is connected to pin 7.

**Note:** Because this is a differential receiver, any handshake or clock signal driving it must be "bi-polar", alternating between a positive voltage and a negative voltage, with respect to the internally grounded negative input. If a device tries to use ground (0 volts) as one of its handshake logic levels, the Macintosh will receive that level as an indeterminate state, with unpredictable results.
The Macintosh Hardware

The SCC itself (at its PCLK pin) is clocked at 3.672 megahertz. The internal synchronous clock (RTxC) pins for both ports are also connected to this 3.672 MHz clock. This is the clock that, after dividing by 16, is normally fed to the SCC's internal baud-rate generator.

The SCC chip generates level-1 processor interrupts during I/O over the serial lines. For more information about SCC interrupts, see chapter 6 of Volume II.

The locations of the SCC control and data lines are given in the following table as offsets from the constant sccWBase for writes, or sccRBase for reads. These base addresses are also available in the global variables SCCWr and SCCRd. The SCC is on the upper byte of the data bus, so you must use only even-addressed byte reads (a byte read of an odd SCC read address tries to reset the entire SCC). When writing, however, you must use only odd-addressed byte writes (the MC68000 puts your data on both bytes of the bus, so it works correctly). A word access to any SCC address will shift the phase of the computer's high-frequency timing by 128 nanoseconds (system software adjusts it correctly during the system startup process).

<table>
<thead>
<tr>
<th>Location</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>sccWBase+aData</td>
<td>Write data register A</td>
</tr>
<tr>
<td>sccRBase+aData</td>
<td>Read data register A</td>
</tr>
<tr>
<td>sccWBase+bData</td>
<td>Write data register B</td>
</tr>
<tr>
<td>sccRBase+bData</td>
<td>Read data register B</td>
</tr>
<tr>
<td>sccWBase+aCtl</td>
<td>Write control register A</td>
</tr>
<tr>
<td>sccRBase+aCtl</td>
<td>Read control register A</td>
</tr>
<tr>
<td>sccWBase+bCtl</td>
<td>Write control register B</td>
</tr>
<tr>
<td>sccRBase+bCtl</td>
<td>Read control register B</td>
</tr>
</tbody>
</table>

Warning: Don't access the SCC chip more often than once every 2.2 µsec. The SCC requires that much time to let its internal lines stabilize.

Refer to the technical specifications of the Zilog Z8530 for the detailed bit maps and control methods (baud rates, protocols, and so on) of the SCC.

Diagram

Figure 4 shows a circuit diagram for the serial ports.

THE MOUSE

The DB-9 connector labeled with the mouse icon connects to the Apple mouse (Apple II, Apple III, Lisa, and Macintosh mice are electrically identical). The mouse generates four square-wave signals that describe the amount and direction of the mouse's travel. Interrupt-driven routines in the Macintosh ROM convert this information into the corresponding motion of the pointer on the screen. By turning an option called mouse scaling on or off in the Control Panel desk accessory, the user can change the amount of screen pointer motion that corresponds to a
Figure 4. Diagram of Serial Ports
The Macintosh Hardware

given mouse motion, depending on how fast the mouse is moved; for more information about mouse scaling, see the discussion of parameter RAM in chapter 13 of Volume II.

**Note:** The mouse is a relative-motion device; that is, it doesn't report where it is, only how far and in which direction it's moving. So if you want to connect graphics tablets, touch screens, light pens, or other absolute-position devices to the mouse port, you must either convert their coordinates into motion information or install your own device-handling routines.

The mouse operates by sending square-wave trains of information to the Macintosh that change as the velocity and direction of motion change. The rubber-coated steel ball in the mouse contacts two capstans, each connected to an interrupter wheel: Motion along the mouse's X axis rotates one of the wheels and motion along the Y axis rotates the other wheel.

The Macintosh uses a scheme known as quadrature to detect which direction the mouse is moving along each axis. There's a row of slots on an interrupter wheel, and two beams of infrared light shine through the slots, each one aimed at a phototransistor detector. The detectors are offset just enough so that, as the wheel turns, they produce two square-wave signals (called the interrupt signal and the quadrature signal) 90 degrees out of phase. The quadrature signal precedes the interrupt signal by 90 degrees when the wheel turns one way, and trails it when the wheel turns the other way.

The interrupt signals, X1 and Y1, are connected to the SCC's DCDA and DCDB inputs, respectively, while the quadrature signals, X2 and Y2, go to inputs of the VIA's data register B. When the Macintosh is interrupted (from the SCC) by the rising edge of a mouse interrupt signal, it checks the VIA for the state of the quadrature signal for that axis: If it's low, the mouse is moving to the left (or down), and if it's high, the mouse is moving to the right (or up). When the SCC interrupts on the falling edge, a high quadrature level indicates motion to the left (or down) and a low quadrature level indicates motion to the right (or up):

<table>
<thead>
<tr>
<th>SCC</th>
<th>VIA</th>
<th>Mouse</th>
</tr>
</thead>
<tbody>
<tr>
<td>X1 (or Y1)</td>
<td>Mouse interrupt</td>
<td>Motion direction in X (or Y) axis</td>
</tr>
<tr>
<td>X2 (or Y2)</td>
<td>Mouse quadrature</td>
<td>Left (or down)</td>
</tr>
<tr>
<td>Low</td>
<td>High</td>
<td>Right (or up)</td>
</tr>
<tr>
<td>Negative edge</td>
<td>Low</td>
<td>Right (or up)</td>
</tr>
<tr>
<td>High</td>
<td></td>
<td>Left (or down)</td>
</tr>
</tbody>
</table>

Figure 5 shows the interrupt (Y1) and quadrature (Y2) signals when the mouse is moved downwards.

The switch on the mouse is a pushbutton that grounds pin 7 on the mouse connector when pressed. The state of the button is checked by software during each vertical blanking interrupt. The small delay between each check is sufficient to debounce the button. You can look directly at the mouse button's state by examining the following bit of VIA data register B (vBase+vBufB):

```assembly
vSW .EQU 3 ;0 = mouse button is down
```

If the bit is clear, the mouse button is down. However, it's recommended that you let the Operating System handle this for you through the event mechanism.

Figure 6 shows the DB-9 pinout for the mouse jack at the back of the Macintosh.
Inside Macintosh

Figure 5. Mouse Mechanism

Figure 6. Pinout for Mouse Jack

Warning: Do not draw more than 200 milliamps at +5 volts from all connectors combined.

Diagram

Figure 7 shows a circuit diagram for the mouse port.

III-28 The Mouse
THE KEYBOARD AND KEYPAD

The Macintosh keyboard and numeric keypad each contain an Intel 8021 microprocessor that scans the keys. The 8021 contains ROM and RAM, and is programmed to conform to the interface protocol described below.

The keyboard plugs into the Macintosh through a four-wire RJ-11 telephone-style jack. If a numeric keypad is installed in the system, the keyboard plugs into it and it in turn plugs into the...
Macintosh. Figure 8 shows the pinout for the keyboard jack on the Macintosh, on the keyboard itself, and on the numeric keypad.

![Pinout for Keyboard Jack](image)

**Figure 8. Pinout for Keyboard Jack**

**Warning:** Do not draw more than 200 milliamps at +5 volts from all connectors combined.

### Keyboard Communication Protocol

The keyboard data line is bidirectional and is driven by whatever device is sending data. The keyboard clock line is driven by the keyboard only. All data transfers are synchronous with the keyboard clock. Each transmission consists of eight bits, with the highest-order bits first.

When sending data to the Macintosh, the keyboard clock transmits eight 330-µsec cycles (160 µsec low, 170 µsec high) on the normally high clock line. It places the data bit on the data line 40 µsec before the falling edge of the clock line and maintains it for 330 µsec. The data bit is clocked into the Macintosh's VIA shift register on the rising edge of the keyboard clock cycle.

When the Macintosh sends data to the keyboard, the keyboard clock transmits eight 400-µsec cycles (180 µsec low, 220 µsec high) on the clock line. On the falling edge of the keyboard clock cycle, the Macintosh places the data bit on the data line and holds it there for 400 µsec. The keyboard reads the data bit 80 µsec after the rising edge of the keyboard clock cycle.

Only the Macintosh can initiate communication over the keyboard lines. On power-up of either the Macintosh or the keyboard, the Macintosh is in charge, and the external device is passive. The Macintosh signals that it's ready to begin communication by pulling the keyboard data line low. Upon detecting this, the keyboard starts clocking and the Macintosh sends a command. The last bit of the command leaves the keyboard data line low; the Macintosh then indicates it's ready to receive the keyboard's response by setting the data line high.

The first command the Macintosh sends out is the Model Number command. The keyboard's response to this command is to reset itself and send back its model number to the Macintosh. If no response is received for 1/2 second, the Macintosh tries the Model Number command again. Once the Macintosh has successfully received a model number from the keyboard, normal
operation can begin. The Macintosh sends the Inquiry command; the keyboard sends back a Key Transition response if a key has been pressed or released. If no key transition has occurred after 1/4 second, the keyboard sends back a Null response to let the Macintosh know it's still there. The Macintosh then sends the Inquiry command again. In normal operation, the Macintosh sends out an Inquiry command every 1/4 second. If it receives no response within 1/2 second, it assumes the keyboard is missing or needs resetting, so it begins again with the Model Number command.

There are two other commands the Macintosh can send: the Instant command, which gets an instant keyboard status without the 1/4-second timeout, and the Test command, to perform a keyboard self-test. Here's a list of the commands that can be sent from the Macintosh to the keyboard:

<table>
<thead>
<tr>
<th>Command name</th>
<th>Value</th>
<th>Keyboard response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inquiry</td>
<td>$10</td>
<td>Key Transition code or Null ($7B)</td>
</tr>
<tr>
<td>Instant</td>
<td>$14</td>
<td>Key Transition code or Null ($7B)</td>
</tr>
<tr>
<td>Model Number</td>
<td>$16</td>
<td>Bit 0: 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bits 1-3: keyboard model number, 1-8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bits 4-6: next device number, 1-8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bit 7: 1 if another device connected</td>
</tr>
<tr>
<td>Test</td>
<td>$36</td>
<td>ACK ($7D) or NAK ($77)</td>
</tr>
</tbody>
</table>

The Key Transition responses are sent out by the keyboard as a single byte: Bit 7 high means a key-up transition, and bit 7 low means a key-down. Bit 0 is always high. The Key Transition responses for key-down transitions on the keyboard are shown (in hexadecimal) in Figure 9. Note that these response codes are different from the key codes returned by the keyboard driver software. The keyboard driver strips off bit 7 of the response and shifts the result one bit to the right, removing bit 0. For example, response code $33 becomes $19, and $2B becomes $15.

Keypad Communication Protocol

When a numeric keypad is used, it must be inserted between the keyboard and the Macintosh; that is, the keypad cable plugs into the jack on the front of the Macintosh, and the keyboard cable plugs into a jack on the numeric keypad. In this configuration, the timings and protocol for the clock and data lines work a little differently: The keypad acts like a keyboard when communicating with the Macintosh, and acts like a Macintosh when communicating over the separate clock and data lines going to the keyboard. All commands from the Macintosh are now received by the keypad instead of the keyboard, and only the keypad can communicate directly with the keyboard.

When the Macintosh sends out an Inquiry command, one of two things may happen, depending on the state of the keypad. If no key transitions have occurred on the keypad since the last Inquiry, the keypad sends an Inquiry command to the keyboard and, later, retransmits the keyboard's response back to the Macintosh. But if a key transition has occurred on the keypad, the keypad responds to an Inquiry by sending back the Keypad response ($79) to the Macintosh. In that case, the Macintosh immediately sends an Instant command, and this time the keypad sends back its own Key Transition response. As with the keyboard, bit 7 high means key-up and bit 7 low means key-down.
The Key Transition responses for key-down transitions on the keypad are shown in Figure 9. Again, note that these response codes are different from the key codes returned by the keyboard driver software. The keyboard driver strips off bit 7 of the response and shifts the result one bit to the right, removing bit 0.
THE DISK INTERFACE

The Macintosh disk interface uses a design similar to that used on the Apple II and Apple III computers, employing the Apple custom IWM chip. Another custom chip called the Analog Signal Generator (ASG) reads the disk speed buffer in RAM and generates voltages that control the disk speed. Together with the VIA, the IWM and the ASG generate all the signals necessary to read, write, format, and eject the 3 1/2-inch disks used by the Macintosh.

The IWM controls four of the disk state-control lines (called CA0, CA1, CA2, and LSTRB), chooses which drive (internal or external) to enable, and processes the disk's read-data and write-data signals. The VIA provides another disk state-control line called SEL.

A buffer in RAM (actually the low-order bytes of words in the sound buffer) is read by the ASG to generate a pulse-width modulated signal that's used to control the speed of the disk motor. The Macintosh Operating System uses this speed control to allow it to store more sectors of information in the tracks closer to the edge of the disk by running the disk motor at slower speeds.

Figure 10 shows the DB-19 pinout for the external disk jack at the back of the Macintosh.

---

Diagram of DB-19 pinout for the external disk jack:

1. Ground
2. Ground
3. Ground
4. Ground
5. -12 volts
6. +5 volts
7. +12 volts
8. +12 volts (not connected)
9. Motor speed control
10. Motor speed control
11. CA0
12. CA1
13. CA2
14. LSTRB
15. Write request
16. SEL
17. External drive enable
18. Read data
19. Write data

---

Warning: This connector was designed for a Macintosh 3 1/2-inch disk drive, which represents a load of 500 milliamps at +12 volts, 500 milliamps at +5 volts, and 0 milliamps at -12 volts. If any other device uses this connector, it must not exceed these loads by more than 100 milliamps at +12 volts, 200 milliamps at +5 volts, and 10 milliamps at -12 volts, including loads from all other connectors combined.
Controlling the Disk State-Control Lines

The IWM contains registers that can be used by the software to control the state-control lines leading out to the disk. By reading or writing certain memory locations, you can turn these state-control lines on or off. Other locations set various IWM internal states. The locations are given in the following table as offsets from the constant dBase, the base address of the IWM; this base address is also available in a global variable named IWM. The IWM is on the lower byte of the data bus, so use odd-addressed byte accesses only.

<table>
<thead>
<tr>
<th>IWM line</th>
<th>Location to turn line on</th>
<th>Location to turn line off</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disk state-control lines:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CA0</td>
<td>dBase+ph0H</td>
<td>dBase+ph0L</td>
</tr>
<tr>
<td>CA1</td>
<td>dBase+ph1H</td>
<td>dBase+ph1L</td>
</tr>
<tr>
<td>CA2</td>
<td>dBase+ph2H</td>
<td>dBase+ph2L</td>
</tr>
<tr>
<td>LSTRB</td>
<td>dBase+ph3H</td>
<td>dBase+ph3L</td>
</tr>
<tr>
<td>Disk enable line:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ENABLE</td>
<td>dBase+motorOn</td>
<td>dBase+motorOff</td>
</tr>
<tr>
<td>IWM internal states:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SELECT</td>
<td>dBase+extDrive</td>
<td>dBase+intDrive</td>
</tr>
<tr>
<td>Q6</td>
<td>dBase+q6H</td>
<td>dBase+q6L</td>
</tr>
<tr>
<td>Q7</td>
<td>dBase+q7H</td>
<td>dBase+q7L</td>
</tr>
</tbody>
</table>

To turn one of the lines on or off, do any kind of memory byte access (read or write) to the respective location.

The CA0, CA1, and CA2 lines are used along with the SEL line from the VIA to select from among the registers and data signals in the disk drive. The LSTRB line is used when writing control information to the disk registers (as described below), and the ENABLE line enables the selected disk drive. SELECT is an IWM internal line that chooses which disk drive can be enabled: On selects the external drive, and off selects the internal drive. The Q6 and Q7 lines are used to set up the internal state of the IWM for reading disk register information, as well as for reading or writing actual disk-storage data.

You can read information from several registers in the disk drive to find out whether the disk is locked, whether a disk is in the drive, whether the head is at track 0, how many heads the drive has, and whether there's a drive connected at all. In turn, you can write to some of these registers to step the head, turn the motor on or off, and eject the disk.

Reading from the Disk Registers

Before you can read from any of the disk registers, you must set up the state of the IWM so that it can pass the data through to the MC68000's memory space where you'll be able to read it. To do that, you must first turn off Q7 by reading or writing dBase+q7L. Then turn on Q6 by accessing dBase+q6H. After that, the IWM will be able to pass data from the disk's RD/SENSE line through to you.

III-34 The Disk Interface
Once you've set up the IWM for disk register access, you must next select which register you want to read. To read one of the disk registers, first enable the drive you want to use (by accessing dBase+intDrive or dBase+extDrive and then dBase+motorOn) and make sure LSTRB is low. Then set CA0, CA1, CA2, and SEL to address the register you want. Once this is done, you can read the disk register data bit in the high-order bit of dBase+q7L. After you've read the data, you may read another disk register by again setting the proper values in CA0, CA1, CA2, and SEL, and then reading dBase+q7L.

Warning: When you're finished reading data from the disk registers, it's important to leave the IWM in a state that the Disk Driver will recognize. To be sure it's in a valid logic state, always turn Q6 back off (by accessing dBase+q6L) after you've finished reading the disk registers.

The following table shows how you must set the disk state-control lines to read from the various disk registers and data signals:

<table>
<thead>
<tr>
<th>State-control lines CA2 CA1 CA0 SEL</th>
<th>Register addressed</th>
<th>Information in register</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0 0</td>
<td>DIRTN</td>
<td>Head step direction</td>
</tr>
<tr>
<td>0 0 0 1</td>
<td>CSTIN</td>
<td>Disk in place</td>
</tr>
<tr>
<td>0 0 1 0</td>
<td>STEP</td>
<td>Disk head stepping</td>
</tr>
<tr>
<td>0 0 1 1</td>
<td>WRTPRT</td>
<td>Disk locked</td>
</tr>
<tr>
<td>0 1 0 0</td>
<td>MOTORON</td>
<td>Disk motor running</td>
</tr>
<tr>
<td>0 1 0 1</td>
<td>TKO</td>
<td>Head at track 0</td>
</tr>
<tr>
<td>0 1 1 1</td>
<td>TACH</td>
<td>Tachometer</td>
</tr>
<tr>
<td>1 0 0 0</td>
<td>RDDATA0</td>
<td>Read data, lower head</td>
</tr>
<tr>
<td>1 0 0 1</td>
<td>RDDATA1</td>
<td>Read data, upper head</td>
</tr>
<tr>
<td>1 1 0 0</td>
<td>SIDES</td>
<td>Single- or double-sided drive</td>
</tr>
<tr>
<td>1 1 1 1</td>
<td>DRVIN</td>
<td>Drive installed</td>
</tr>
</tbody>
</table>

Writing to the Disk Registers

To write to a disk register, first be sure that LSTRB is off, then turn on CA0 and CA1. Next, set SEL to 0. Set CA0 and CA1 to the proper values from the table below, then set CA2 to the value you want to write to the disk register. Hold LSTRB high for at least one µsec but not more than one msec (unless you're ejecting a disk) and bring it low again. Be sure that you don't change CA0-CA2 or SEL while LSTRB is high, and that CA0 and CA1 are set high before changing SEL.

The following table shows how you must set the disk state-control lines to write to the various disk registers:

<table>
<thead>
<tr>
<th>Control lines CA1 CA0 SEL</th>
<th>Register addressed</th>
<th>Register function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0</td>
<td>DIRTN</td>
<td>Set stepping direction</td>
</tr>
<tr>
<td>0 1 0</td>
<td>STEP</td>
<td>Step disk head one track</td>
</tr>
<tr>
<td>1 0 0</td>
<td>MOTORON</td>
<td>Turn on/off disk motor</td>
</tr>
<tr>
<td>1 1 0</td>
<td>EJECT</td>
<td>Eject the disk</td>
</tr>
</tbody>
</table>
Explanations of the Disk Registers

The information written to or read from the various disk registers can be interpreted as follows:

- The DIRTN signal sets the direction of subsequent head stepping: 0 causes steps to go toward the inside track (track 79), 1 causes them to go toward the outside track (track 0).
- CSTIN is 0 only when a disk is in the drive.
- Setting STEP to 0 steps the head one full track in the direction last set by DIRTN. When the step is complete (about 12 msec), the disk drive sets STEP back to 1, and then you can step again.
- WRTPRT is 0 whenever the disk is locked. Do not write to a disk unless WRTPRT is 1.
- MOTORON controls the state of the disk motor: 0 turns on the motor, and 1 turns it off. The motor will run only if the drive is enabled and a disk is in place; otherwise, writing to this line will have no effect.
- TKO goes to 0 only if the head is at track 0. This is valid beginning 12 msec after the step that puts it at track 0.
- Writing 1 to EJECT ejects the disk from the drive. To eject a disk, you must hold LSTRB high for at least 1/2 second.
- The current disk speed is available as a pulse train on TACH. The TACH line produces 60 pulses for each rotation of the drive motor. The disk motor speed is controlled by the ASG as it reads the disk speed RAM buffer.
- RDDATA0 and RDDATA1 carry the instantaneous data from the disk head.
- SIDES is always 0 on single-sided drives and 1 on double-sided drives.
- DRVIN is always 0 if the selected disk drive is physically connected to the Macintosh, otherwise it floats to 1.

THE REAL-TIME CLOCK

The Macintosh real-time clock is a custom chip whose interface lines are available through the VIA. The clock contains a four-byte counter that's incremented once each second, as well as a line that can be used by the VIA to generate an interrupt once each second. It also contains 20 bytes of RAM that are powered by a battery when the Macintosh is turned off. These RAM bytes, called parameter RAM, contain important data that needs to be preserved even when the system power is not available. The Operating System maintains a copy of parameter RAM that you can access in low memory. To find out how to use the values in parameter RAM, see chapter 13 of Volume II.
The Macintosh Hardware

Accessing the Clock Chip

The clock is accessed through the following bits of VIA data register B (vBase+vBufB):

- rTCData .EQU 0 ;real-time clock serial data line
- rTCClk .EQU 1 ;real-time clock data-clock line
- rTCEnb .EQU 2 ;real-time clock serial enable

These three bits constitute a simple serial interface. The rTCData bit is a bidirectional serial data line used to send command and data bytes back and forth. The rTCClk bit is a data-clock line, always driven by the processor (you set it high or low yourself) that regulates the transmission of the data and command bits. The rTCEnb bit is the serial enable line, which signals the real-time clock that the processor is about to send it serial commands and data.

To access the clock chip, you must first enable its serial function. To do this, set the serial enable line (rTCEnb) to 0. Keep the serial enable line low during the entire transaction; if you set it to 1, you'll abort the transfer.

Warning: Be sure you don't alter any of bits 3-7 of VIA data register B during clock serial access.

A command can be either a write request or a read request. After the eight bits of a write request, the clock will expect the next eight bits across the serial data line to be your data for storage into one of the internal registers of the clock. After receiving the eight bits of a read request, the clock will respond by putting eight bits of its data on the serial data line. Commands and data are transferred serially in eight-bit groups over the serial data line, with the high-order bit first and the low-order bit last.

To send a command to the clock, first set the rTCData bit of VIA data direction register B (vBase+vDirB) so that the real-time clock's serial data line will be used for output to the clock. Next, set the rTCClk bit of vBase+vBufB to 0, then set the rTCData bit to the value of the first (high-order) bit of your data byte. Then raise (set to 1) the data-clock bit (rTCClk). Then lower the data-clock, set the serial data line to the next bit, and raise the data-clock line again. After the last bit of your command has been sent in this way, you can either continue by sending your data byte in the same way (if your command was a write request) or switch to receiving a data byte from the clock (if your command was a read request).

To receive a byte of data from the clock, you must first send a command that's a read request. After you've clocked out the last bit of the command, clear the rTCData bit of the data direction register so that the real-time clock's serial data line can be used for input from the clock; then lower the data-clock bit (rTCClk) and read the first (high-order) bit of the clock's data byte on the serial data line. Then raise the data-clock, lower it again, and read the next bit of data. Continue this until all eight bits are read, then raise the serial enable line (rTCEnb), disabling the data transfer.

The following table lists the commands you can send to the clock. A 1 in the high-order bit makes your command a read request; a 0 in the high-order bit makes your command a write request. (In this table, "z" is the bit that determines read or write status, and bits marked "a" are bits whose values depend on what parameter RAM byte you want to address.)
### Inside Macintosh

<table>
<thead>
<tr>
<th>Command byte</th>
<th>Register addressed by the command</th>
</tr>
</thead>
<tbody>
<tr>
<td>z0000001</td>
<td>Seconds register 0 (lowest-order byte)</td>
</tr>
<tr>
<td>z0000101</td>
<td>Seconds register 1</td>
</tr>
<tr>
<td>z0001001</td>
<td>Seconds register 2</td>
</tr>
<tr>
<td>z0001101</td>
<td>Seconds register 3 (highest-order byte)</td>
</tr>
<tr>
<td>00110001</td>
<td>Test register (write only)</td>
</tr>
<tr>
<td>00110101</td>
<td>Write-protect register (write only)</td>
</tr>
<tr>
<td>z010aa01</td>
<td>RAM address 100aa ($10-$13)</td>
</tr>
<tr>
<td>z1aaaa01</td>
<td>RAM address 0aaaa ($00-$0F)</td>
</tr>
</tbody>
</table>

Note that the last two bits of a command byte must always be 01.

If the high-order bit (bit 7) of the write-protect register is set, this prevents writing into any other register on the clock chip (including parameter RAM). Clearing the bit allows you to change any values in any registers on the chip. Don't try to read from this register; it's a write-only register.

The two highest-order bits (bits 7 and 6) of the test register are used as device control bits during testing, and should always be set to 0 during normal operation. Setting them to anything else will interfere with normal clock counting. Like the write-protect register, this is a write-only register; don't try to read from it.

All clock data must be sent as full eight-bit bytes, even if only one or two bits are of interest. The rest of the bits may not matter, but you must send them to the clock or the write will be aborted when you raise the serial enable line.

It's important to use the proper sequence if you're writing to the clock's seconds registers. If you write to a given seconds register, there's a chance that the clock may increment the data in the next higher-order register during the write, causing unpredictable results. To avoid this possibility, always write to the registers in low-to-high order. Similarly, the clock data may increment during a read of all four time bytes, which could cause invalid data to be read. To avoid this, always read the time twice (or until you get the same value twice).

**Warning:** When you've finished reading from the clock registers, always end by doing a final write such as setting the write-protect bit. Failure to do this may leave the clock in a state that will run down the battery more quickly than necessary.

### The One-Second Interrupt

The clock also generates a VIA interrupt once each second (if this interrupt is enabled). The enable status for this interrupt can be read from or written to bit 0 of the VIA's interrupt enable register (vBase+vIER). When reading the enable register, a 1 bit indicates the interrupt is enabled, and 0 means it's disabled. Writing $01 to the enable register disables the clock's one-second interrupt (without affecting any other interrupts), while writing $81 enables it again. See chapter 6 of Volume II for more information about writing your own interrupt handlers.

**Warning:** Be sure when you write to bit 0 of the VIA's interrupt enable register that you don't change any of the other bits.
The Synertek SY6522 Versatile Interface Adapter (VIA) controls the keyboard, internal real-time clock, parts of the disk, sound, and mouse interfaces, and various internal Macintosh signals. Its base address is available as the constant vBase and is also stored in a global variable named VIA. The VIA is on the upper byte of the data bus, so use even-addressed byte accesses only.

There are two parallel data registers within the VIA, called A and B, each with a data direction register. There are also several event timers, a clocked shift register, and an interrupt flag register with an interrupt enable register.

Normally you won't have to touch the direction registers, since the Operating System sets them up for you at system startup. A 1 bit in a data direction register means the corresponding bit of the respective data register will be used for output, while a 0 bit means it will be used for input.

Note: For more information on the registers and control structure of the VIA, consult the technical specifications for the SY6522 chip.

### VIA Register A

VIA data register A is at vBase+vBufA. The corresponding data direction register is at vBase+vDirA.

<table>
<thead>
<tr>
<th>Bit(s)</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>vSCCWReq</td>
<td>SCC wait/request</td>
</tr>
<tr>
<td>6</td>
<td>vPage2</td>
<td>Alternate screen buffer</td>
</tr>
<tr>
<td>5</td>
<td>vHeadSel</td>
<td>Disk SEL line</td>
</tr>
<tr>
<td>4</td>
<td>vOverlay</td>
<td>ROM low-memory overlay</td>
</tr>
<tr>
<td>3</td>
<td>vSndPg2</td>
<td>Alternate sound buffer</td>
</tr>
<tr>
<td>0-2</td>
<td>vSound (mask)</td>
<td>Sound volume</td>
</tr>
</tbody>
</table>

The vSCCWReq bit can signal that the SCC has received a character (used to maintain serial communications during disk accesses, when the CPU's interrupts from the SCC are disabled). The vPage2 bit controls which screen buffer is being displayed, and the vHeadSel bit is the SEL control line used by the disk interface. The vOverlay bit (used only during system startup) can be used to place another image of ROM at the bottom of memory, where RAM usually is (RAM moves to $600000). The sound buffer is selected by the vSndPg2 bit. Finally, the vSound bits control the sound volume.

### VIA Register B

VIA data register B is at vBase+vBufB. The corresponding data direction register is at vBase+vDirB.
Inside Macintosh

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>vSndEnb</td>
<td>Sound enable/disable</td>
</tr>
<tr>
<td>6</td>
<td>vH4</td>
<td>Horizontal blanking</td>
</tr>
<tr>
<td>5</td>
<td>vY2</td>
<td>Mouse Y2</td>
</tr>
<tr>
<td>4</td>
<td>vX2</td>
<td>Mouse X2</td>
</tr>
<tr>
<td>3</td>
<td>vSW</td>
<td>Mouse switch</td>
</tr>
<tr>
<td>2</td>
<td>rTCEnb</td>
<td>Real-time clock serial enable</td>
</tr>
<tr>
<td>1</td>
<td>rTCClk</td>
<td>Real-time clock data-clock line</td>
</tr>
<tr>
<td>0</td>
<td>rTCData</td>
<td>Real-time clock serial data</td>
</tr>
</tbody>
</table>

The vSndEnb bit turns the sound generator on or off, and the vH4 bit is set when the video beam is in its horizontal blanking period. The vY2 and vX2 bits read the quadrature signals from the Y (vertical) and X (horizontal) directions, respectively, of the mouse's motion lines. The vSW bit reads the mouse switch. The rTCEnb, rTCClk, and rTCData bits control and read the real-time clock.

The VIA Peripheral Control Register

The VIA's peripheral control register, at vBase+vPCR, allows you to set some very low-level parameters (such as positive-edge or negative-edge triggering) dealing with the keyboard data and clock interrupts, the one-second real-time clock interrupt line, and the vertical blanking interrupt.

<table>
<thead>
<tr>
<th>Bit(s)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>5-7</td>
<td>Keyboard data interrupt control</td>
</tr>
<tr>
<td>4</td>
<td>Keyboard clock interrupt control</td>
</tr>
<tr>
<td>1-3</td>
<td>One-second interrupt control</td>
</tr>
<tr>
<td>0</td>
<td>Vertical blanking interrupt control</td>
</tr>
</tbody>
</table>

The VIA Timers

The timers controlled by the VIA are called timer 1 and timer 2. Timer 1 is used to time various events having to do with the Macintosh sound generator. Timer 2 is used by the Disk Driver to time disk I/O events. If either timer isn't being used by the Operating System, you're free to use it for your own purposes. When a timer counts down to 0, an interrupt will be generated if the proper interrupt enable has been set. See chapter 6 of Volume II for information about writing your own interrupt handlers.

To start one of the timers, store the appropriate values in the high- and low-order bytes of the timer counter (or the timer 1 latches, for multiple use of the value). The counters and latches are at the following locations:
Location | Contents
---|---
vBase+vT1C | Timer 1 counter (low-order byte)
vBase+vT1CH | Timer 1 counter (high-order byte)
vBase+vT1L | Timer 1 latch (low-order byte)
vBase+vT1LH | Timer 1 latch (high-order byte)
vBase+vT2C | Timer 2 counter (low-order byte)
vBase+vT2CH | Timer 2 counter (high-order byte)

Note: When setting a timer, it's not enough to simply store a full word to the high-order address, because the high- and low-order bytes of the counters are not adjacent. You must explicitly do two stores, one for the high-order byte and one for the low-order byte.

**VIA Interrupts**

The VIA (through its IRQ line) can cause a level-0 processor interrupt whenever one of the following occurs: Timer 1 or timer 2 times out; the keyboard is clocking a bit in through its serial port; the shift register for the keyboard serial interface has finished shifting in or out; the vertical blanking interval is beginning; or the one-second clock has ticked. For more information on how to use these interrupts, see chapter 6 of Volume II.

The interrupt flag register at vBase+vIFR contains flag bits that are set whenever the interrupt corresponding to that bit has occurred. The Operating System uses these flags to determine which device has caused an interrupt. Bit 7 of the interrupt flag register is not really a flag: It remains set (and the IRQ line to the processor is held low) as long as any enabled VIA interrupt is occurring.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Interrupting device</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>IRQ (all enabled VIA interrupts)</td>
</tr>
<tr>
<td>6</td>
<td>Timer 1</td>
</tr>
<tr>
<td>5</td>
<td>Timer 2</td>
</tr>
<tr>
<td>4</td>
<td>Keyboard clock</td>
</tr>
<tr>
<td>3</td>
<td>Keyboard data bit</td>
</tr>
<tr>
<td>2</td>
<td>Keyboard data ready</td>
</tr>
<tr>
<td>1</td>
<td>Vertical blanking interrupt</td>
</tr>
<tr>
<td>0</td>
<td>One-second interrupt</td>
</tr>
</tbody>
</table>

The interrupt enable register, at vBase+vIER, lets you enable or disable any of these interrupts. If an interrupt is disabled, its bit in the interrupt flag register will continue to be set whenever that interrupt occurs, but it won't affect the IRQ flag, nor will it interrupt the processor.

The bits in the interrupt enable register are arranged just like those in the interrupt flag register, except for bit 7. When you write to the interrupt enable register, bit 7 is "enable/disable": If bit 7 is a 1, each 1 in bits 0-6 enables the corresponding interrupt; if bit 7 is a 0, each 1 in bits 0-6 disables that interrupt. In either case, 0's in bits 0-6 do not change the status of those interrupts. Bit 7 is always read as a 1.
Inside Macintosh

Other VIA Registers

The shift register, at vBase+vSR, contains the eight bits of data that have been shifted in or that will be shifted out over the keyboard data line.

The auxiliary control register, at vBase+vACR, is described in the SY6522 documentation. It controls various parameters having to do with the timers and the shift register.

SYSTEM STARTUP

When power is first supplied to the Macintosh, a carefully orchestrated sequence of events takes place.

First, the processor is held in a wait state while a series of circuits gets the system ready for operation. The VIA and IWM are initialized, and the mapping of ROM and RAM are altered temporarily by setting the overlay bit in VIA data register A. This places the ROM starting at the normal ROM location $400000$, and a duplicate image of the same ROM starting at address 0 (where RAM normally is), while RAM is placed starting at $600000$. Under this mapping, the Macintosh software executes out of the normal ROM locations above $400000$, but the MC68000 can obtain some critical low-memory vectors from the ROM image it finds at address 0.

Next, a memory test and several other system tests take place. After the system is fully tested and initialized, the software clears the VIA's overlay bit, mapping the system RAM back where it belongs, starting at address 0. Then the disk startup process begins.

First the internal disk is checked: If there's a disk inserted, the system attempts to read it. If no disk is in the internal drive and there's an external drive with an inserted disk, the system will try to read that one. Otherwise, the question-mark disk icon is displayed until a disk is inserted. If the disk startup fails for some reason, the "sad Macintosh" icon is displayed and the Macintosh goes into an endless loop until it's turned off again.

Once a readable disk has been inserted, the first two sectors (containing the system startup blocks) are read in and the normal disk load begins.
SUMMARY

Warning: This information applies only to the Macintosh 128K and 512K, not to the Macintosh XL.

Constants

; VIA base addresses
vBase .EQU $EFE1FE ;main base for VIA chip (in variable VIA)
aVBufB .EQU vBase ;register B base
aVBufA .EQU $EFFFFE ;register A base
aVBufM .EQU aVBufB ;register containing mouse signals
aVIFR .EQU $EFFBFE ;interrupt flag register
aVIER .EQU $EFFDFE ;interrupt enable register

; Offsets from vBase
vBufB .EQU 512*0 ;register B (zero offset)
vDirB .EQU 512*2 ;register B direction register
vDirA .EQU 512*3 ;register A direction register
vT1C .EQU 512*4 ;timer 1 counter (low-order byte)
vT1CH .EQU 512*5 ;timer 1 counter (high-order byte)
vT1L .EQU 512*6 ;timer 1 latch (low-order byte)
vT1LH .EQU 512*7 ;timer 1 latch (high-order byte)
vT2C .EQU 512*8 ;timer 2 counter (low-order byte)
vT2CH .EQU 512*9 ;timer 2 counter (high-order byte)
vSR .EQU 512*10 ;shift register (keyboard)
vACR .EQU 512*11 ;auxiliary control register
vPCR .EQU 512*12 ;peripheral control register
vIFR .EQU 512*13 ;interrupt flag register
vIER .EQU 512*14 ;interrupt enable register
vBufA .EQU 512*15 ;register A

; VIA register A constants
vAOut .EQU $7F ;direction register A: 1 bits = outputs
vAINit .EQU $7B ;initial value for vBufA (medium volume)
vSound .EQU 7 ;sound volume bits

; VIA register A bit numbers
vSndPg2 .EQU 3 ;0 = alternate sound buffer
vOverlay .EQU 4 ;1 = ROM overlay (system startup only)
vHeadSel .EQU 5 ;disk SEL control line
vPage2 .EQU 6 ;0 = alternate screen buffer
vSCCWReq .EQU 7 ;SCC wait/request line
Inside Macintosh

; VIA register B constants
vBOut .EQU $87 ; direction register B: 1 bits = outputs
vBInit .EQU $07 ; initial value for vBufB

; VIA register B bit numbers
rTCDATA .EQU 0 ; real-time clock serial data line
rTCClk .EQU 1 ; real-time clock data-clock line
rTCEnb .EQU 2 ; real-time clock serial enable
vSW .EQU 3 ; 0 = mouse button is down
vX2 .EQU 4 ; mouse X quadrature level
vY2 .EQU 5 ; mouse Y quadrature level
vH4 .EQU 6 ; 1 = horizontal blanking
vSndEnb .EQU 7 ; 0 = sound enabled, 1 = disabled

; SCC base addresses
sccRBase .EQU $9FFFF8 ; SCC base read address (in variable SCCRd)
sccWBase .EQU $BFFFF9 ; SCC base write address (in variable SCCWr)

; Offsets from SCC base addresses
aData .EQU 6 ; channel A data in or out
aCtl .EQU 2 ; channel A control
bData .EQU 4 ; channel B data in or out
bCtl .EQU 0 ; channel B control

; Bit numbers for control register RR0
rxBF .EQU 0 ; 1 = SCC receive buffer full
txBE .EQU 2 ; 1 = SCC send buffer empty

; IWM base address
dBase .EQU $DFE1FF ; IWM base address (in variable IWM)

; Offsets from dBase
phOL .EQU 512*0 ; CA0 off (0)
phOH .EQU 512*1 ; CA0 on (1)
phlL .EQU 512*2 ; CA1 off (0)
phlH .EQU 512*3 ; CA1 on (1)
ph2L .EQU 512*4 ; CA2 off (0)
ph2H .EQU 512*5 ; CA2 on (1)
ph3L .EQU 512*6 ; LSTRB off (low)
ph3H .EQU 512*7 ; LSTRB on (high)
mtrOff .EQU 512*8 ; disk enable off
mtrOn .EQU 512*9 ; disk enable on
intDrive .EQU 512*10 ; select internal drive
extDrive .EQU 512*11 ; select external drive
q6L .EQU 512*12 ; Q6 off

III-44 Summary
The Macintosh Hardware

| q6H | .EQU 512*13 | ;Q6 on |
| q7L | .EQU 512*14 | ;Q7 off |
| q7H | .EQU 512*15 | ;Q7 on |

; Screen and sound addresses for 512K Macintosh (will also work for 128K, since addresses wrap)

| screenLow  | .EQU $7A700 | ;top left corner of main screen buffer |
| soundLow   | .EQU $7FD00 | ;main sound buffer (in variable SoundBase) |
| pwmBuffer  | .EQU $7FD01 | ;main disk speed buffer |
| ovlyRAM    | .EQU $600000 | ;RAM start address when overlay is set |
| ovlyScreen | .EQU $67A700 | ;screen start with overlay set |
| romStart   | .EQU $400000 | ;ROM start address (in variable ROMBase) |

Variables

| ROMBase     | Base address of ROM |
| SoundBase   | Address of main sound buffer |
| SCCRd       | SCC read base address |
| SCCWr       | SCC write base address |
| IWM         | IWM base address |
| VIA         | VIA base address |

Exception Vectors

<table>
<thead>
<tr>
<th>Location</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>$00</td>
<td>Reset: initial stack pointer (not a vector)</td>
</tr>
<tr>
<td>$04</td>
<td>Reset: initial vector</td>
</tr>
<tr>
<td>$08</td>
<td>Bus error</td>
</tr>
<tr>
<td>$0C</td>
<td>Address error</td>
</tr>
<tr>
<td>$10</td>
<td>Illegal instruction</td>
</tr>
<tr>
<td>$14</td>
<td>Divide by zero</td>
</tr>
<tr>
<td>$18</td>
<td>CHK instruction</td>
</tr>
<tr>
<td>$1C</td>
<td>TRAPV instruction</td>
</tr>
<tr>
<td>$20</td>
<td>Privilege violation</td>
</tr>
<tr>
<td>$24</td>
<td>Trace interrupt</td>
</tr>
<tr>
<td>$28</td>
<td>Line 1010 emulator</td>
</tr>
<tr>
<td>$2C</td>
<td>Line 1111 emulator</td>
</tr>
<tr>
<td>$30-$3B</td>
<td>Unassigned (reserved)</td>
</tr>
<tr>
<td>$3C</td>
<td>Uninitialized interrupt</td>
</tr>
<tr>
<td>$40-$5F</td>
<td>Unassigned (reserved)</td>
</tr>
</tbody>
</table>
### Summary

<table>
<thead>
<tr>
<th>Location</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>$60</td>
<td>Spurious interrupt</td>
</tr>
<tr>
<td>$64</td>
<td>VIA interrupt</td>
</tr>
<tr>
<td>$68</td>
<td>SCC interrupt</td>
</tr>
<tr>
<td>$6C</td>
<td>VIA+SCC vector (temporary)</td>
</tr>
<tr>
<td>$70</td>
<td>Interrupt switch</td>
</tr>
<tr>
<td>$74</td>
<td>Interrupt switch + VIA</td>
</tr>
<tr>
<td>$78</td>
<td>Interrupt switch + SCC</td>
</tr>
<tr>
<td>$7C</td>
<td>Interrupt switch + VIA + SCC</td>
</tr>
<tr>
<td>$80-$BF</td>
<td>TRAP instructions</td>
</tr>
<tr>
<td>$C0-$FF</td>
<td>Unassigned (reserved)</td>
</tr>
</tbody>
</table>
### SUMMARY

<table>
<thead>
<tr>
<th>Page</th>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>49</td>
<td>3</td>
<td>About This Chapter</td>
</tr>
<tr>
<td>50</td>
<td>3</td>
<td>AppleTalk Manager</td>
</tr>
<tr>
<td>65</td>
<td>3</td>
<td>Binary-Decimal Conversion Package</td>
</tr>
<tr>
<td>66</td>
<td>3</td>
<td>Control Manager</td>
</tr>
<tr>
<td>71</td>
<td>3</td>
<td>Desk Manager</td>
</tr>
<tr>
<td>73</td>
<td>3</td>
<td>Device Manager</td>
</tr>
<tr>
<td>80</td>
<td>3</td>
<td>Dialog Manager</td>
</tr>
<tr>
<td>85</td>
<td>3</td>
<td>Disk Driver</td>
</tr>
<tr>
<td>88</td>
<td>3</td>
<td>Disk Initialization Package</td>
</tr>
<tr>
<td>90</td>
<td>3</td>
<td>Event Manager, Operating System</td>
</tr>
<tr>
<td>94</td>
<td>3</td>
<td>Event Manager, Toolbox</td>
</tr>
<tr>
<td>98</td>
<td>3</td>
<td>File Manager</td>
</tr>
<tr>
<td>113</td>
<td>3</td>
<td>Font Manager</td>
</tr>
<tr>
<td>118</td>
<td>3</td>
<td>International Utilities Package</td>
</tr>
<tr>
<td>124</td>
<td>3</td>
<td>Memory Manager</td>
</tr>
<tr>
<td>130</td>
<td>3</td>
<td>Menu Manager</td>
</tr>
<tr>
<td>134</td>
<td>3</td>
<td>Package Manager</td>
</tr>
<tr>
<td>135</td>
<td>3</td>
<td>Printing Manager</td>
</tr>
<tr>
<td>141</td>
<td>3</td>
<td>QuickDraw</td>
</tr>
<tr>
<td>154</td>
<td>3</td>
<td>Resource Manager</td>
</tr>
<tr>
<td>157</td>
<td>3</td>
<td>Scrap Manager</td>
</tr>
<tr>
<td>159</td>
<td>3</td>
<td>Segment Loader</td>
</tr>
<tr>
<td>161</td>
<td>3</td>
<td>Serial Drivers</td>
</tr>
<tr>
<td>165</td>
<td>3</td>
<td>Sound Driver</td>
</tr>
<tr>
<td>172</td>
<td>3</td>
<td>Standard File Package</td>
</tr>
<tr>
<td>176</td>
<td>3</td>
<td>System Error Handler</td>
</tr>
<tr>
<td>179</td>
<td>3</td>
<td>TextEdit</td>
</tr>
<tr>
<td>184</td>
<td>3</td>
<td>Utilities, Operating System</td>
</tr>
<tr>
<td>190</td>
<td>3</td>
<td>Utilities, Toolbox</td>
</tr>
<tr>
<td>193</td>
<td>3</td>
<td>Vertical Retrace Manager</td>
</tr>
<tr>
<td>195</td>
<td>3</td>
<td>Window Manager</td>
</tr>
<tr>
<td>201</td>
<td>3</td>
<td>Assembly Language</td>
</tr>
<tr>
<td>201</td>
<td>3</td>
<td>Miscellaneous Variables</td>
</tr>
<tr>
<td>201</td>
<td>3</td>
<td>Hardware</td>
</tr>
</tbody>
</table>
ABOUT THIS CHAPTER

This chapter includes all the summaries that appear at the end of other chapters of *Inside Macintosh*. The summaries are arranged in alphabetical order of the part of the Toolbox or Operating System being summarized.

Note: The summaries of the Event Managers are listed under "Event Manager, Operating System" and "Event Manager, Toolbox". The Toolbox and Operating System Utilities are listed similarly.

The last section of this chapter, "Assembly Language", contains information for assembly-language programmers only. It lists some miscellaneous global variables along with hardware-related definitions for the Macintosh 128K and 512K.
APPLETALK MANAGER

Constants

CONST lapSize = 20; {ABusRecord size for ALAP}
    ddpSize = 26; {ABusRecord size for DDP}
    npbSize = 26; {ABusRecord size for NBP}
    atpSize = 56; {ABusRecord size for ATP}

Data Types

TYPE ABProtoType = (lapProto, ddpProto, npbProto, atpProto);

ABRecHandle = ^ABRecPtr;
ABRecPtr = ^ABusRecord;
ABusRecord = RECORD
    abOpcode: ABCallType; {type of call}
    abResult: INTEGER; {result code}
    abUserReference: LONGINT; {for your use}
CASE ABProtoType OF
    lapProto:
        (lapAddress: LAPAddrBlock; {destination or source node ID}
         lapReqCount: INTEGER; {length of frame data or buffer }
                        { size in bytes}
         lapActCount: INTEGER; {number of frame data bytes }
                        { actually received}
         lapDataPtr: Pte); {pointer to frame data or pointer }
                        { to buffer}
    ddpProto:
        (ddpType: Byte; {DDP protocol type}
         ddpSocket: Byte; {source or listening socket number}
         ddpAddress: AddrBlock; {destination or source socket address}
         ddpReqCount: INTEGER; {length of datagram data or buffer }
                        { size in bytes}
         ddpActCount: INTEGER; {number of bytes actually received}
         ddpDataPtr: Pte; {pointer to buffer}
         ddpNodeID: Byte); {original destination node ID}
    npbProto:
        (nbpEntityPtr: EntityPtr; {pointer to entity name}
         npbBufPtr: Pte; {pointer to buffer}
         npbBufSize: INTEGER; {buffer size in bytes}
         npbDataField: INTEGER; {number of addresses or }
                        { socket number}
         npbAddress: AddrBlock; {socket address}
         npbRetransmitInfo: RetransType); {retransmission information}
Summary

atpProto:
(atpSocket: Byte; {listening or responding socket}
  (number)}
atpAddress: AddrBlock; {destination or source socket}
  (address)}
atpReqCount: INTEGER; {request size or buffer size}
atpDataPtr: Ptr; {pointer to buffer}
atpRspBDSPtr: BDSptr; {pointer to response BDS}
atpBitMap: BitMapType; {transaction bit map}
atpTransID: INTEGER; {transaction ID}
atpActCount: INTEGER; {number of bytes actually received}
atpUserData: LONGINT; {user bytes}
atpXO: BOOLEAN; {exactly-once flag}
atpEOM: BOOLEAN; {end-of-message flag}
atpTimeOut: Byte; {retry timeout interval in seconds}
atpRetries: Byte; {maximum number of retries}
atpNumBufs: Byte; {number of elements in response}
  (BDS or number of response packets)
  (packets sent)
atpNumRsp: Byte; {number of response packets}
  (received or sequence number)
atpBDSSize: Byte; {number of elements in response BDS}
atpRepUData: LONGINT; {user bytes sent or received in transaction response}
atpRepBuf: Ptr; {pointer to response message buffer}
atpRepSize: INTEGER); {size of response message buffer}

END;

ABCAllType = (tLAPRead, tLAPWrite, tDDPRead, tDDPWrite, tNBPLookup,
  tNBPConfirm, tNBPRegister, tATPSndRequest,
  tATPGetRequest, tATPSdRsp, tATPAddRsp, tATPRequest,
  tATPResponse);

LAPAdrBlock = PACKED RECORD
  dstNodeID: Byte; {destination node ID}
  srcNodeID: Byte; {source node ID}
  lapProtType: AByte {ALAP protocol type}
END;

ABByte = 1..127; {ALAP protocol type}

AddrBlock = PACKED RECORD
  aNet: INTEGER; {network number}
  aNode: Byte; {node ID}
  aSocket: Byte {socket number}
END;

BDSptr = ^BDSType;
BDSType = ARRAY[0..7] OF BDSElement; {response BDS}
Inside Macintosh

BDSElement = RECORD
    buffSize: INTEGER; {buffer size in bytes}
    buffPtr: Ptr; {pointer to buffer}
    dataSize: INTEGER; {number of bytes actually received}
    userBytes: LONGINT {user bytes}
END;

BitMapType = PACKED ARRAY[0..7] OF BOOLEAN;

EntityPtr = "EntityName;
EntityName = RECORD
    objStr: Str32; {object}
    typeStr: Str32; {type}
    zoneStr: Str32 {zone}
END;

Str32 = STRING[32];

RetransType =
    PACKED RECORD
        retransInterval: Byte; {retransmit interval in 8-tick units}
        retransCount: Byte {total number of attempts}
    END;

Routines  [Not in ROM]

Opening and Closing AppleTalk

FUNCTION MPPOpen : OSErr;
FUNCTION MPPClose : OSErr;

AppleTalk Link Access Protocol

FUNCTION LAPOpenProtocol (theLAPType: ABByte; protoPtr: Ptr) : OSErr;
FUNCTION LAPCloseProtocol (theLAPType: ABByte) : OSErr;
FUNCTION LAPWrite (abRecord: ABRecHandle; async: BOOLEAN) : OSErr;
← abOpcode {always tLAPWrite}
← abResult {result code}
→ abUserReference {for your use}
→ lapAddress.dstNodeID {destination node ID}
→ lapAddress.lapProtType {ALAP protocol type}
→ lapReqCount {length of frame data}
→ lapDataPtr {pointer to frame data}
FUNCTION LAPRead (abRecord: ABRecHandle; async: BOOLEAN) : OSErr;
    ← abOpcode {always tLAPRead}
    ← abResult {result code}
    → abUserReference {for your use}
    ← lapAddress.dstNodeID {destination node ID}
    ← lapAddress.srcNodeID {source node ID}
    → lapAddress.lapProtType {ALAP protocol type}
    ← lapReqCount {buffer size in bytes}
    ← lapActCount {number of frame data bytes actually received}
    → lapDataPtr {pointer to buffer}

FUNCTION LAPRdCancel (abRecord: ABRecHandle) : OSErr;

Datagram Delivery Protocol

FUNCTION DDPOpenSocket (VAR theSocket: Byte; sktListener: Ptr) : OSErr;
FUNCTION DDPCloseSocket (theSocket: Byte) : OSErr;

FUNCTION DDPWrite (abRecord: ABRecHandle; doChecksurn: BOOLEAN; async: BOOLEAN) : OSErr;
    ← abOpcode {always tDDPWrite}
    ← abResult {result code}
    → abUserReference {for your use}
    ← ddpType {DDP protocol type}
    ← ddpSocket {source socket number}
    ← ddpAddress {destination socket address}
    ← ddpReqCount {length of datagram data}
    → ddpDataPtr {pointer to buffer}

FUNCTION DDPRead (abRecord: ABRecHandle; retCksumErrs: BOOLEAN; async: BOOLEAN) : OSErr;
    ← abOpcode {always tDDPRead}
    ← abResult {result code}
    → abUserReference {for your use}
    ← ddpType {DDP protocol type}
    ← ddpSocket {listening socket number}
    ← ddpAddress {source socket address}
    ← ddpReqCount {buffer size in bytes}
    ← ddpActCount {number of bytes actually received}
    → ddpDataPtr {pointer to buffer}
    ← ddpNodeID {original destination node ID}

FUNCTION DDPRdCancel (abRecord: ABRecHandle) : OSErr;

AppleTalk Transaction Protocol

FUNCTION ATPLoad : OSErr;
FUNCTION ATPUload : OSErr;
FUNCTION ATPOpenSocket (addrRcvd: AddrBlock; VAR atpSocket: Byte) : OSErr;
FUNCTION ATPCloseSocket (atpSocket: Byte) : OSErr;
FUNCTION ATPSndRequest (abRecord: ABRecHandle; async: BOOLEAN) : OSErr;

← abOpcode {always tATPSndRequest}
← abResult {result code}
→ abUserReference {for your use}
→ atpAddress {destination socket address}
→ atpReqCount {request size in bytes}
→ atpDataPtr {pointer to buffer}
→ atpRspBDSPtr {pointer to response BDS}
→ atpUserData {user bytes}
→ atpXO {exactly-once flag}
← atpEOM {end-of-message flag}
→ atpTimeOut {retry timeout interval in seconds}
→ atpRetries {maximum number of retries}
→ atpNumBufs {number of elements in response BDS}
← atpNumRsp {number of response packets actually received}

FUNCTION ATPRequest (abRecord: ABRecHandle; async: BOOLEAN) : OSErr;

← abOpcode {always tATPRequest}
← abResult {result code}
→ abUserReference {for your use}
→ atpAddress {destination socket address}
→ atpReqCount {request size in bytes}
→ atpDataPtr {pointer to buffer}
← atpActCount {number of bytes actually received}
→ atpUserData {user bytes}
→ atpXO {exactly-once flag}
← atpEOM {end-of-message flag}
→ atpTimeOut {retry timeout interval in seconds}
→ atpRetries {maximum number of retries}
← atpRspUserData {user bytes received in transaction response}
→ atpRspBuf {pointer to response message buffer}
→ atpRspSize {size of response message buffer}

FUNCTION ATPReqCancel (abRecord: ABRecHandle; async: BOOLEAN) : OSErr;

FUNCTION ATPGetRequest (abRecord: ABRecHandle; async: BOOLEAN) : OSErr;

← abOpcode {always tATPGetRequest}
← abResult {result code}
→ abUserReference {for your use}
→ atpSocket {listening socket number}
→ atpAddress {source socket address}
→ atpReqCount {buffer size in bytes}
→ atpDataPtr {pointer to buffer}
→ atpBitMap {transaction bit map}
→ atpTransID {transaction ID}
← atpActCount {number of bytes actually received}
← atpUserData {user bytes}
← atpXO {exactly-once flag}
FUNCTION ATPSndRsp (abRecord: ABRecHandle; async: BOOLEAN) : OSErr;
  abOpcode  {always tATPSndRsp}
  abResult  {result code}
  abUserReference {for your use}
  atpSocket  {responding socket number}
  atpAddress {destination socket address}
  atpRspBDSPtr {pointer to response BDS}
  atpTransID {transaction ID}
  atpEOM {end-of-message flag}
  atpNumBufs {number of response packets being sent}
  atpBDSSize {number of elements in response BDS}

FUNCTION ATPAddRsp (abRecord: ABRecHandle) : OSErr;
  abOpcode  {always tATPAddRsp}
  abResult  {result code}
  abUserReference {for your use}
  atpSocket  {responding socket number}
  atpAddress {destination socket address}
  atpReqCount {buffer size in bytes}
  atpDataPtr {pointer to buffer}
  atpTransID {transaction ID}
  atpUserData {user bytes}
  atpEOM {end-of-message flag}
  atpNumRsp {sequence number}

FUNCTION ATPResponse (abRecord: ABRecHandle; async: BOOLEAN) : OSErr;
  abOpcode  {always tATPResponse}
  abResult  {result code}
  abUserReference {for your use}
  atpSocket  {responding socket number}
  atpAddress {destination socket address}
  atpTransID {transaction ID}
  atpRspUData {user bytes sent in transaction response}
  atpRspBuf {pointer to response message buffer}
  atpRspSize {size of response message buffer}

FUNCTION ATPRspCancel (abRecord: ABRecHandle; async: BOOLEAN) : OSErr;

Name-Binding Protocol

FUNCTION NBPRegister (abRecord: ABRecHandle; async: BOOLEAN) : OSErr;
  abOpcode  {always tNBPRegister}
  abResult  {result code}
  abUserReference {for your use}
  nbpEntityPtr {pointer to entity name}
  nbpBufPtr {pointer to buffer}
  nbpBufSize {buffer size in bytes}
  nbpAddress.aSocket {socket address}
  nbpRetransmitInfo {retransmission information}
FUNCTION NBPLookup (abRecord: ABRecHandle; async: BOOLEAN) : OSErr;

← abOpcode {always tNBPLookup}
← abResult {result code}
→ abUserReference {for your use}
→ nbpEntityPtr {pointer to entity name}
→ nbpBufPtr {pointer to buffer}
→ nbpBufSize {buffer size in bytes}
← nbpDataField {number of addresses received}
→ nbpRetransmitInfo {retransmission information}

FUNCTION NBPExtract (theBuffer: Ptr; numInBuf: INTEGER; whichOne: INTEGER; VAR abEntity: EntityName; VAR address: AddrBlock) : OSErr;

FUNCTION NBPConfirm (abRecord: ABRecHandle; async: BOOLEAN) : OSErr;

← abOpcode {always tNBPConfirm}
← abResult {result code}
→ abUserReference {for your use}
→ nbpEntityPtr {pointer to entity name}
← nbpDataField {socket number}
→ nbpAddress {socket address}
→ nbpRetransmitInfo {retransmission information}

FUNCTION NBPRemove (abEntity: EntityPtr) : OSErr;
FUNCTION NBPLoad : OSErr;
FUNCTION NBPUnload : OSErr;

Miscellaneous Routines

FUNCTION GetNodeAddress (VAR myNode, myNet: INTEGER) : OSErr;
FUNCTION IsMPPOpen : BOOLEAN;
FUNCTION IsATPOpen : BOOLEAN;

Result Codes

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>atpBadRsp</td>
<td>-3107</td>
<td>Bad response from ATPRequest</td>
</tr>
<tr>
<td>atpLenErr</td>
<td>-3106</td>
<td>ATP response message too large</td>
</tr>
<tr>
<td>badATPSkt</td>
<td>-1099</td>
<td>ATP bad responding socket</td>
</tr>
<tr>
<td>badBuffNum</td>
<td>-1100</td>
<td>ATP bad sequence number</td>
</tr>
<tr>
<td>buf2SmallErr</td>
<td>-3101</td>
<td>ALAP frame too large for buffer</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DDP datagram too large for buffer</td>
</tr>
<tr>
<td>cbNotFound</td>
<td>-1102</td>
<td>ATP control block not found</td>
</tr>
<tr>
<td>cksumErr</td>
<td>-3103</td>
<td>DDP bad checksum</td>
</tr>
<tr>
<td>ddpLenErr</td>
<td>-92</td>
<td>DDP datagram or ALAP data length too big</td>
</tr>
<tr>
<td>Name</td>
<td>Value</td>
<td>Meaning</td>
</tr>
<tr>
<td>-------------------</td>
<td>-------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ddpSktErr</td>
<td>-91</td>
<td>DDP socket error: socket already active; not a well-known socket; socket table full; all dynamic socket numbers in use</td>
</tr>
<tr>
<td>excessCollsns</td>
<td>-95</td>
<td>ALAP no CTS received after 32 RTS's, or line sensed in use 32 times (not necessarily caused by collisions)</td>
</tr>
<tr>
<td>extractErr</td>
<td>-3104</td>
<td>NBP can't find tuple in buffer</td>
</tr>
<tr>
<td>lapProtErr</td>
<td>-94</td>
<td>ALAP error attaching/detaching ALAP protocol type: attach error when ALAP protocol type is negative, not in range, already in table, or when table is full; detach error when ALAP protocol type isn't in table</td>
</tr>
<tr>
<td>npBuffOvr</td>
<td>-1024</td>
<td>NBP buffer overflow</td>
</tr>
<tr>
<td>npConfDiff</td>
<td>-1026</td>
<td>NBP name confirmed for different socket</td>
</tr>
<tr>
<td>npDuplicate</td>
<td>-1027</td>
<td>NBP duplicate name already exists</td>
</tr>
<tr>
<td>npNISErr</td>
<td>-1029</td>
<td>NBP names information socket error</td>
</tr>
<tr>
<td>npNoConfirm</td>
<td>-1025</td>
<td>NBP name not confirmed</td>
</tr>
<tr>
<td>npNotFound</td>
<td>-1028</td>
<td>NBP name not found</td>
</tr>
<tr>
<td>noBridgeErr</td>
<td>-93</td>
<td>No bridge found</td>
</tr>
<tr>
<td>noDataArea</td>
<td>-1104</td>
<td>Too many outstanding ATP calls</td>
</tr>
<tr>
<td>noErr</td>
<td>0</td>
<td>No error</td>
</tr>
<tr>
<td>noMPPError</td>
<td>-3102</td>
<td>MPP driver not installed</td>
</tr>
<tr>
<td>noRelErr</td>
<td>-1101</td>
<td>ATP no release received</td>
</tr>
<tr>
<td>noSendResp</td>
<td>-1103</td>
<td>ATPAcpAddRsp issued before ATPSndRsp</td>
</tr>
<tr>
<td>portInUse</td>
<td>-97</td>
<td>Driver Open error, port already in use</td>
</tr>
<tr>
<td>portNotCfg</td>
<td>-98</td>
<td>Driver Open error, port not configured for this connection</td>
</tr>
<tr>
<td>readQErr</td>
<td>-3105</td>
<td>Socket or protocol type invalid or not found in table</td>
</tr>
<tr>
<td>recNotFnd</td>
<td>-3108</td>
<td>ABRecord not found</td>
</tr>
<tr>
<td>reqAborted</td>
<td>-1105</td>
<td>Request aborted</td>
</tr>
<tr>
<td>reqFailed</td>
<td>-1096</td>
<td>ATPSndRsp failed: retry count exceeded</td>
</tr>
<tr>
<td>sktClosedErr</td>
<td>-3109</td>
<td>Asynchronous call aborted because socket was closed before call was completed</td>
</tr>
<tr>
<td>tooManyReqs</td>
<td>-1097</td>
<td>ATP too many concurrent requests</td>
</tr>
<tr>
<td>tooManySkts</td>
<td>-1098</td>
<td>ATP too many responding sockets</td>
</tr>
</tbody>
</table>

*AppleTalk Manager III-57*
Assembly-Language Information

Constants

; Serial port use types
useFree .EQU 0 ; unconfigured
useATalk .EQU 1 ; configured for AppleTalk
useASync .EQU 2 ; configured for the Serial Driver

; Bit in PortBUse for .ATP driver status
atpLoadedBit .EQU 4 ; set if .ATP driver is opened

; Unit numbers for AppleTalk drivers
mppUnitNum .EQU 9 ; .MPP driver
atpUnitNum .EQU 10 ; .ATP driver

; csCode values for Control calls (MPP)
writeLAP .EQU 243
detachPH .EQU 244
attachPH .EQU 245
writeDDP .EQU 246
closeSkt .EQU 247
openSkt .EQU 248
loadNBP .EQU 249
confirmName .EQU 250
lookupName .EQU 251
removeName .EQU 252
registerName .EQU 253
killNBP .EQU 254
unloadNBP .EQU 255

; csCode values for Control calls (ATP)
relRspCB .EQU 249
closeATPSkt .EQU 250
addResponse .EQU 251
sendResponse .EQU 252
getRequest .EQU 253
openATPSkt .EQU 254
sendRequest .EQU 255
relTCB .EQU 256

; ALAP header
lapDstAdr .EQU 0 ; destination node ID
lapSrcAdr .EQU 1 ; source node ID
lapType .EQU 2 ; ALAP protocol type

III-58 AppleTalk Manager
; ALAP header size
lapHdSz .EQU 3

; ALAP protocol type values
shortDDP .EQU 1 ;short DDP header
longDDP .EQU 2 ;long DDP header

; Long DDP header
ddpHopCnt .EQU 0 ;count of bridges passed (4 bits)
ddpLength .EQU 0 ;datagram length (10 bits)
ddpChecksum .EQU 2 ;checksum
ddpDstNet .EQU 4 ;destination network number
ddpSrcNet .EQU 6 ;source network number
ddpDstNode .EQU 8 ;destination node ID
ddpSrcNode .EQU 9 ;source node ID
ddpDstSkt .EQU 10 ;destination socket number
ddpSrcSkt .EQU 11 ;source socket number
ddpType .EQU 12 ;DDP protocol type

; DDP long header size
ddpHSzLong .EQU ddpType+1

; Short DDP header
ddpLength .EQU 0 ;datagram length
sDDPDstSkt .EQU ddpChecksum ;destination socket number
sDDPSrcSkt .EQU sDDPDstSkt+1 ;source socket number
sDDPType .EQU sDDPSrcSkt+1 ;DDP protocol type

; DDP short header size
ddpHSzShort .EQU sDDPType+1

; Mask for datagram length
ddpLenMask .EQU $03FF

; Maximum size of DDP data
ddpMaxData .EQU 586

; ATP header
atpControl .EQU 0 ;control information
atpBitMap .EQU 1 ;bit map
atpRespNo .EQU 1 ;sequence number
atpTransID .EQU 2 ;transaction ID
atpUserData .EQU 4 ;user bytes
Inside Macintosh

; ATP header size
atpHdSz .EQU 8

; DDP protocol type for ATP packets
atp .EQU 3

; ATP function code
atpReqCode .EQU $40 ; TReq packet
atpRspCode .EQU $80 ; TResp packet
atpRelCode .EQU $C0 ; TRel packet

; ATPFlags control information bits
sendChk .EQU 0 ; send-checksum bit
tidValid .EQU 1 ; transaction ID validity bit
atpSTSBit .EQU 3 ; send-transmission-status bit
atpEOMBit .EQU 4 ; end-of-message bit
atpXOBit .EQU 5 ; exactly-once bit

; Maximum number of ATP request packets
atpMaxNum .EQU 8

; ATP buffer data structure
bdsBuffSz .EQU 0 ; size of data to send or buffer size
bdsBuffAddr .EQU 2 ; pointer to data or buffer
bdsDataSz .EQU 6 ; number of bytes actually received
bdsUserData .EQU 8 ; user bytes

; BDS element size
bdsEntrySz .EQU 12

; NBP packet
nbpControl .EQU 0 ; packet type
nbpCnt .EQU 0 ; tuple count
nbpID .EQU 1 ; packet identifier
nbpTuple .EQU 2 ; start of first tuple

; DDP protocol type for NBP packets
nbp .EQU 2

III-60 AppleTalk Manager
; NBP packet types
brRq .EQU 1 ; broadcast request
lkUp .EQU 2 ; lookup request
lkUpReply .EQU 3 ; lookup reply

; NBP tuple
tupleNet .EQU 0 ; network number
tupleNode .EQU 2 ; node ID
tupleSkt .EQU 3 ; socket number
tupleEnum .EQU 4 ; used internally
tupleName .EQU 5 ; entity name

; Maximum number of tuples in NBP packet
tupleMax .EQU 15

; NBP meta-characters
equals .EQU '=' ; "wild-card" meta-character
star .EQU '*'; "this zone" meta-character

; NBP names table entry
ntLink .EQU 0 ; pointer to next entry
ntTuple . EQU 4 ; tuple
ntSocket . EQU 7 ; socket number
ntEntity . EQU 9 ; entity name

; NBP names information socket number
nis . EQU 2

Routines

Link Access Protocol

WriteLAP function
→ 26 csCode word ; always writeLAP
→ 30 wdsPointer pointer ; write data structure

AttachPH function
→ 26 csCode word ; always attachPH
→ 28 protType byte ; ALAP protocol type
→ 30 handler pointer ; protocol handler
Inside Macintosh

DetachPH function

→ 26 csCode word ;always detachPH
→ 28 protType byte ;ALAP protocol type

Datagram Delivery Protocol

OpenSkt function

→ 26 csCode word ;always openSkt
↔ 28 socket byte ;socket number
→ 30 listener pointer ;socket listener

CloseSkt function

→ 26 csCode word ;always closeSkt
→ 28 socket byte ;socket number

WriteDDP function

→ 26 csCode word ;always writeDDP
→ 28 socket byte ;socket number
→ 29 checksumFlag byte ;checksum flag
→ 30 wdsPointer pointer ;write data structure

AppleTalk Transaction Protocol

OpenATPSkt function

→ 26 csCode word ;always openATPSkt
↔ 28 atpSocket byte ;socket number
→ 30 addrBlock long word ;socket request specification

CloseATPSkt function

→ 26 csCode word ;always closeATPSkt
→ 28 atpSocket byte ;socket number

SendRequest function

→ 18 userData long word ;user bytes
↔ 22 reqTID word ;transaction ID used in request
→ 26 csCode word ;always sendRequest
↔ 28 currBitMap byte ;bit map
↔ 29 atpFlags byte ;control information
→ 30 addrBlock long word ;destination socket address
→ 34 reqLength word ;request size in bytes
→ 36 reqPointer pointer ;pointer to request data
→ 40 bdsPointer pointer ;pointer to response BDS
→ 44 numOfBuffers byte ;number of responses expected
→ 45 timeoutVal byte ;timeout interval
↔ 46 numOfResps byte ;number of responses received
↔ 47 retryCount byte ;number of retries

GetRequest function

← 18 userData long word ;user bytes
→ 26 csCode word ;always getRequest
→ 28 atpSocket byte ;socket number

III-62 AppleTalk Manager
Summary

GetRequest function

\[
\begin{align*}
\leftarrow & \quad 18 \quad \text{userData} & \text{long word} & \text{user bytes} \\
\rightarrow & \quad 26 \quad \text{csCode} & \text{word} & \text{always getRequest} \\
\rightarrow & \quad 28 \quad \text{atpSocket} & \text{byte} & \text{socket number} \\
\leftarrow & \quad 29 \quad \text{atpFlags} & \text{byte} & \text{control information} \\
\leftarrow & \quad 30 \quad \text{addrBlock} & \text{long word} & \text{source of request} \\
\leftarrow & \quad 34 \quad \text{reqLength} & \text{word} & \text{request buffer size} \\
\leftarrow & \quad 36 \quad \text{reqPointer} & \text{pointer} & \text{pointer to request buffer} \\
\leftarrow & \quad 44 \quad \text{bitMap} & \text{byte} & \text{bit map} \\
\leftarrow & \quad 46 \quad \text{transID} & \text{word} & \text{transaction ID} \\
\end{align*}
\]

SendResponse function

\[
\begin{align*}
\leftarrow & \quad 18 \quad \text{userData} & \text{long word} & \text{user bytes from TRel} \\
\rightarrow & \quad 26 \quad \text{csCode} & \text{word} & \text{always sendResponse} \\
\rightarrow & \quad 28 \quad \text{atpSocket} & \text{byte} & \text{socket number} \\
\rightarrow & \quad 29 \quad \text{atpFlags} & \text{byte} & \text{control information} \\
\rightarrow & \quad 30 \quad \text{addrBlock} & \text{long word} & \text{response destination} \\
\rightarrow & \quad 40 \quad \text{bdsPointer} & \text{pointer} & \text{pointer to response BDS} \\
\rightarrow & \quad 44 \quad \text{numOfBuffs} & \text{byte} & \text{number of response packets being sent} \\
\rightarrow & \quad 45 \quad \text{bdsSize} & \text{byte} & \text{BDS size in elements} \\
\rightarrow & \quad 46 \quad \text{transID} & \text{word} & \text{transaction ID} \\
\end{align*}
\]

AddResponse function

\[
\begin{align*}
\rightarrow & \quad 18 \quad \text{userData} & \text{long word} & \text{user bytes} \\
\rightarrow & \quad 26 \quad \text{csCode} & \text{word} & \text{always addResponse} \\
\rightarrow & \quad 28 \quad \text{atpSocket} & \text{byte} & \text{socket number} \\
\rightarrow & \quad 29 \quad \text{atpFlags} & \text{byte} & \text{control information} \\
\rightarrow & \quad 30 \quad \text{addrBlock} & \text{long word} & \text{response destination} \\
\rightarrow & \quad 34 \quad \text{reqLength} & \text{word} & \text{response size} \\
\rightarrow & \quad 36 \quad \text{reqPointer} & \text{pointer} & \text{pointer to response} \\
\rightarrow & \quad 44 \quad \text{rspNum} & \text{byte} & \text{sequence number} \\
\rightarrow & \quad 46 \quad \text{transID} & \text{word} & \text{transaction ID} \\
\end{align*}
\]

RelITCB function

\[
\begin{align*}
\rightarrow & \quad 26 \quad \text{csCode} & \text{word} & \text{always relITCB} \\
\rightarrow & \quad 30 \quad \text{addrBlock} & \text{long word} & \text{destination of request} \\
\rightarrow & \quad 46 \quad \text{transID} & \text{word} & \text{transaction ID of request} \\
\end{align*}
\]

RelRspCB function

\[
\begin{align*}
\rightarrow & \quad 26 \quad \text{csCode} & \text{word} & \text{always relRspCB} \\
\rightarrow & \quad 28 \quad \text{atpSocket} & \text{byte} & \text{socket number that request was received on} \\
\rightarrow & \quad 30 \quad \text{addrBlock} & \text{long word} & \text{source of request} \\
\rightarrow & \quad 46 \quad \text{transID} & \text{word} & \text{transaction ID of request} \\
\end{align*}
\]
Name-Binding Protocol

RegisterName function
- → 26 csCode word ;always registerName
- → 28 interval byte ;retry interval
- ← 29 count byte ;retry count
- → 30 ntQEIPtr pointer ;names table element pointer
- → 34 verifyFlag byte ;set if verify needed

LookupName function
- → 26 csCode word ;always lookupName
- → 28 interval byte ;retry interval
- ← 29 count byte ;retry count
- → 30 entityPtr pointer ;pointer to entity name
- → 34 retBuffPtr pointer ;pointer to buffer
- → 38 retBuffSize word ;buffer size in bytes
- → 40 maxToGet word ;matches to get
- ← 42 numGotten word ;matches found

ConfirmName function
- → 26 csCode word ;always confirmName
- → 28 interval byte ;retry interval
- ← 29 count byte ;retry count
- → 30 entityPtr pointer ;pointer to entity name
- → 34 confirmAddr pointer ;entity address
- ← 38 newSocket byte ;socket number

RemoveName function
- → 26 csCode word ;always removeName
- → 30 entityPtr pointer ;pointer to entity name

LoadNBP function
- → 26 csCode word ;always loadNBP

UnloadNBP function
- → 26 csCode word ;always unloadNBP

Variables

SPConfig Use types for serial ports (byte)
(bits 0-3: current configuration of serial port B
bits 4-6: current configuration of serial port A)

PortBUse Current availability of serial port B (byte)
(bit 7: 1 = not in use, 0 = in use
bits 0-3: current use of port bits
bits 4-6: driver-specific)

ABusVars Pointer to AppleTalk variables
BINARY-DECIMAL CONVERSION PACKAGE

Routines

PROCEDURE NumToString (theNum: LONGINT; VAR theString: Str255);
PROCEDURE StringToNum (theString: Str255; VAR theNum: LONGINT);

Assembly-Language Information

Constants

; Routine selectors
numToString .EQU 0
stringToNum .EQU 1

Routines

<table>
<thead>
<tr>
<th>Name</th>
<th>On entry</th>
<th>On exit</th>
</tr>
</thead>
<tbody>
<tr>
<td>NumToString</td>
<td>A0: ptr to theString (preceded by length byte)</td>
<td>A0: ptr to theString</td>
</tr>
<tr>
<td></td>
<td>D0: theNum (long)</td>
<td></td>
</tr>
<tr>
<td>StringToNum</td>
<td>A0: ptr to theString (preceded by length byte)</td>
<td>D0: theNum (long)</td>
</tr>
</tbody>
</table>

Trap Macro Name

_Pack7
CONTROL MANAGER

Constants

CONST { Control definition IDs }

pushButProc = 0; {simple button}
checkBoxProc = 1; {check box}
radioButProc = 2; {radio button}
useWFont = 8; {add to above to use window's font}
scrollBarProc = 16; {scroll bar}

{ Part codes }
inButton = 10; {simple button}
inCheckBox = 11; {check box or radio button}
inUpButton = 20; {up arrow of a scroll bar}
inDownButton = 21; {down arrow of a scroll bar}
inPageUp = 22; {"page up" region of a scroll bar}
inPageDown = 23; {"page down" region of a scroll bar}
inThumb = 129; {thumb of a scroll bar}

{ Axis constraints for DragControl }

noConstraint = 0; {no constraint}
hAxisOnly = 1; {horizontal axis only}
vAxisOnly = 2; {vertical axis only}

{ Messages to control definition function }

drawCnt1 = 0; {draw the control (or control part)}
testCnt1 = 1; {test where mouse button was pressed}
calcCRgns = 2; {calculate control's region (or indicator's)}
initCnt1 = 3; {do any additional control initialization}
dispCnt1 = 4; {take any additional disposal actions}
posCnt1 = 5; {reposition control's indicator and update it}
thumbCnt1 = 6; {calculate parameters for dragging indicator}
dragCnt1 = 7; {drag control (or its indicator)}
autoTrack = 8; {execute control's action procedure}

Data Types

TYPE ControlHandle = ^ControlPtr;
ControlPtr = ^ControlRecord;
ControlRecord =
PACKED RECORD
  nextControl: ControlHandle; {next control}
  controlOwner: WindowPtr; {control's window}
  controlRect: Rect; {enclosing rectangle}
  controlVis: Byte; {255 if visible}
  controlHilite: Byte; [highlight state]
  controlValue: INTEGER; {control's current setting}
  controlMin: INTEGER; {control's minimum setting}
  controlMax: INTEGER; {control's maximum setting}
  controlDefProc: Handle; {control definition function}
  controlData: Handle; {data used by controlDefProc}
  controlAction: ProcPtr; {default action procedure}
  controlRefCon: LONGINT; {control's reference value}
  controlTitle: Str255 {control's title}
END;

Routines

Initialization and Allocation

FUNCTION NewControl (theWindow: WindowPtr; boundsRect: Rect; title: Str255; visible: BOOLEAN; value: INTEGER; min, max: INTEGER; procID: INTEGER; refCon: LONGINT) : ControlHandle;

FUNCTION GetNewControl (controlID: INTEGER; theWindow: WindowPtr) : ControlHandle;

PROCEDURE DisposeControl (theControl: ControlHandle);

PROCEDURE KillControls (theWindow: WindowPtr);

Control Display

PROCEDURE SetCTitle (theControl: ControlHandle; title: Str255);

PROCEDURE GetCTitle (theControl: ControlHandle; VAR title: Str255);

PROCEDURE HideControl (theControl: ControlHandle);

PROCEDURE ShowControl (theControl: ControlHandle);

PROCEDURE DrawControls (theWindow: WindowPtr);

PROCEDURE HiliteControl (theControl: ControlHandle; hiliteState: INTEGER);

Mouse Location

FUNCTION FindControl (thePoint: Point; theWindow: WindowPtr; VAR whichControl: ControlHandle) : INTEGER;

FUNCTION TrackControl (theControl: ControlHandle; startPt: Point; actionProc: ProcPtr) : INTEGER;

FUNCTION TestControl (theControl: ControlHandle; thePoint: Point) : INTEGER;
Control Movement and Sizing

PROCEDURE MoveControl (theControl: ControlHandle; h,v: INTEGER);
PROCEDURE DragControl (theControl: ControlHandle; startPt: Point;
    limitRect,slopRect: Rect; axis: INTEGER);
PROCEDURE SizeControl (theControl: ControlHandle; w,h: INTEGER);

Control Setting and Range

PROCEDURE SetCtlValue (theControl: ControlHandle; theValue: INTEGER);
FUNCTION GetCtlValue (theControl: ControlHandle) : INTEGER;
PROCEDURE SetCtlMin (theControl: ControlHandle; minValue: INTEGER);
FUNCTION GetCtlMin (theControl: ControlHandle) : INTEGER;
PROCEDURE SetCtlMax (theControl: ControlHandle; maxValue: INTEGER);
FUNCTION GetCtlMax (theControl: ControlHandle) : INTEGER;

Miscellaneous Routines

PROCEDURE SetCRefCon (theControl: ControlHandle; data: LONGINT);
FUNCTION GetCRefCon (theControl: ControlHandle) : LONGINT;
PROCEDURE SetCtlAction (theControl: ControlHandle; actionProc ProcPtr);
FUNCTION GetCtlAction (theControl: ControlHandle) : ProcPtr;

Action Procedure for TrackControl

If an indicator: PROCEDURE MyAction;
If not an indicator: PROCEDURE MyAction (theControl: ControlHandle;
    partCode: INTEGER);

Control Definition Function

FUNCTION MyControl (varCode: INTEGER; theControl: ControlHandle;
    message: INTEGER; param: LONGINT) : LONGINT;

Assembly-Language Information

Constants

; Control definition IDs
pushButProc .EQU 0 ; simple button
checkBoxProc .EQU 1 ; check box
radioButProc .EQU 2 ; radio button
useWFont .EQU 8 ; add to above to use window's font
scrollBarProc .EQU 16 ; scroll bar
; Part codes
inButton     .EQU 10    ; simple button
inCheckBox   .EQU 11    ; check box or radio button
inUpButton   .EQU 20    ; up arrow of a scroll bar
inDownButton .EQU 21    ; down arrow of a scroll bar
inPageUp     .EQU 22    ; "page up" region of a scroll bar
inPageDown   .EQU 23    ; "page down" region of a scroll bar
inThumb      .EQU 129   ; thumb of a scroll bar

; Axis constraints for DragControl
noConstraint .EQU 0     ; no constraint
hAxisOnly    .EQU 1     ; horizontal axis only
vAxisOnly    .EQU 2     ; vertical axis only

; Messages to control definition function
drawCtlMsg   .EQU 0     ; draw the control (or control part)
hitCtlMsg    .EQU 1     ; test where mouse button was pressed
calcCtlMsg   .EQU 2     ; calculate control's region (or indicator's)
newCtlMsg    .EQU 3     ; do any additional control initialization
dispCtlMsg   .EQU 4     ; take any additional disposal actions
posCtlMsg    .EQU 5     ; reposition control's indicator and update it
thumbCtlMsg  .EQU 6     ; calculate parameters for dragging indicator
dragCtlMsg   .EQU 7     ; drag control (or its indicator)
trackCtlMsg  .EQU 8     ; execute control's action procedure

Control Record Data Structure

nextControl  Handle to next control in control list
contrOwner    Pointer to this control's window
contrRect     Control's enclosing rectangle (8 bytes)
contrVis      255 if control is visible (byte)
contrHilite   Highlight state (byte)
contrValue    Control's current setting (word)
contrMin      Control's minimum setting (word)
contrMax      Control's maximum setting (word)
contrDefHandle Handle to control definition function
contrData     Data used by control definition function (long)
contrAction   Address of default action procedure
contrRfCon    Control's reference value (long)
contrTitle    Handle to control's title (preceded by length byte)
contrSize     Size in bytes of control record except contrTitle field
Inside Macintosh

**Special Macro Names**

<table>
<thead>
<tr>
<th>Pascal name</th>
<th>Macro name</th>
</tr>
</thead>
<tbody>
<tr>
<td>DisposeControl</td>
<td>_DisposControl</td>
</tr>
<tr>
<td>GetCtlMax</td>
<td>_GetMaxCtl</td>
</tr>
<tr>
<td>GetCtlMin</td>
<td>_GetMinCtl</td>
</tr>
<tr>
<td>SetCtlMax</td>
<td>_SetMaxCtl</td>
</tr>
<tr>
<td>SetCtlMin</td>
<td>_SetMinCtl</td>
</tr>
</tbody>
</table>

**Variables**

<table>
<thead>
<tr>
<th>DragHook</th>
<th>Address of procedure to execute during TrackControl and DragControl</th>
</tr>
</thead>
<tbody>
<tr>
<td>DragPattern</td>
<td>Pattern of dragged region's outline (8 bytes)</td>
</tr>
</tbody>
</table>
**DESK MANAGER**

**Routines**

**Opening and Closing Desk Accessories**

FUNCTION OpenDeskAcc (theAcc: Str255) : INTEGER;
PROCEDURE CloseDeskAcc (refNum: INTEGER);

**Handling Events in Desk Accessories**

PROCEDURE SystemClick (theEvent: EventRecord; theWindow: WindowPtr);
FUNCTION SystemEdit (editCmd: INTEGER) : BOOLEAN;

**Performing Periodic Actions**

PROCEDURE SystemTask;

**Advanced Routines**

FUNCTION SystemEvent (theEvent: EventRecord) : BOOLEAN;
PROCEDURE SystemMenu (menuResult: LONGINT);

**Assembly-Language Information**

**Constants**

; Desk accessory flag
dNeedTime .EQU 5  ;set if driver needs time for performing a
                ; periodic action

; Control routine messages
accEvent .EQU 64  ;handle a given event
accRun .EQU 65   ;take the periodic action, if any, for
                 ; this desk accessory
accCursor .EQU 66  ;change cursor shape if appropriate;
                  ; generate null event if window was
                  ; created by Dialog Manager
accMenu .EQU 67   ;handle a given menu item
accUndo .EQU 68   ;handle the Undo command
Inside Macintosh

accCut .EQU 70 ;handle the Cut command
accCopy .EQU 71 ;handle the Copy command
accPaste .EQU 72 ;handle the Paste command
accClear .EQU 73 ;handle the Clear command

Special Macro Names

<table>
<thead>
<tr>
<th>Pascal name</th>
<th>Macro name</th>
</tr>
</thead>
<tbody>
<tr>
<td>SystemEdit</td>
<td>_SysEdit</td>
</tr>
</tbody>
</table>

Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MBarEnable</td>
<td>Unique menu ID for active desk accessory, when menu bar belongs to the</td>
</tr>
<tr>
<td></td>
<td>accessory (word)</td>
</tr>
<tr>
<td>SEvtEnb</td>
<td>0 if SystemEvent should return FALSE (byte)</td>
</tr>
</tbody>
</table>
DEVICE MANAGER

Constants

CONST { Values for requesting read/write access }

fsCurPerm  = 0; {whatever is currently allowed}
fsRdPerm   = 1; {request to read only}
fsWrPerm   = 2; {request to write only}
fsRdWrPerm = 3; {request to read and write}

{ Positioning modes }

fsAtMark    = 0; {at current position}
fsFromStart = 1; {offset relative to beginning of medium}
fsFromMark  = 3; {offset relative to current position}
rDVerify    = 64; {add to above for read-verify}

Data Types

TYPE ParamBlkType = (ioParam, fileParam, volumeParam, cntrlParam);

ParmBlkPtr = 'ParamBlockRec;
ParamBlockRec = RECORD
  qLink:  QElemPtr; {next queue entry}
  qType:  INTEGER; {queue type}
  ioTrap: INTEGER; {routine trap}
  ioCmdAddr: Ptr; {routine address}
  ioCompletion: ProcPtr; {completion routine}
  ioResult: OSErr; {result code}
  ioNamePtr: StringPtr; {driver name}
  i0VRefNum: INTEGER; {volume reference or drive number}
CASE ParamBlkType OF
  ioParam:
    {ioRefNum: INTEGER; {driver reference number}
    ioVersNum: SignedByte; {not used}
    ioPermssn: SignedByte; {read/write permission}
    ioMisc:  Ptr; {not used}
    ioBuffer: Ptr; {pointer to data buffer}
    ioReqCount: LONGINT; {requested number of bytes}
    ioActCount: LONGINT; {actual number of bytes}
    ioPosMode: INTEGER; {positioning mode}
    ioPosOffset: LONGINT; {positioning offset}
  fileParam:
    . . . {used by File Manager}
  volumeParam:
    . . . {used by File Manager}
Inside Macintosh

cntrlParam:
  (ioCRefNum: INTEGER; {driver reference number})
  csCode: INTEGER; {type of Control or Status call}
  csParam: ARRAY[0..10] OF INTEGER) {control or status information}
END;

DCtlHandle = ^DCtlPtr;
DCtlPtr = ^DCtlEntry;
DCtlEntry =
RECORD
  dCtlDriver: Ptr; {pointer to ROM driver or handle to } [ RAM driver]
  dCtlFlags: INTEGER; {flags}
  dCtlQHdr: QHdr; {driver I/O queue header}
  dCtlPosition: LONGINT; {byte position used by Read and } 
  { Write calls}
  dCtlStorage: Handle; {[ handle to RAM driver's private } 
  { storage]}
  dCtlRefNum: INTEGER; {driver reference number}
  dCtlCurTicks: LONGINT; {used internally}
  dCtlWindow: WindowPtr; {pointer to driver's window}
  dCtlDelay: INTEGER; {number of ticks between periodic } 
  { actions]
  dCtlEMask: INTEGER; {desk accessory event mask}
  dCtlMenu: INTEGER {menu ID of menu associated with } 
  { driver]
END;

High-Level Routines [Not in ROM]

FUNCTION OpenDriver (name: Str255; VAR refNum: INTEGER) : OSErr;
FUNCTION CloseDriver (refNum: INTEGER) : OSErr;
FUNCTION FSRead (refNum: INTEGER; VAR count: LONGINT; buffPtr: Ptr) : OSErr;
FUNCTION FSWrite (refNum: INTEGER; VAR count: LONGINT; buffPtr: Ptr) : OSErr;
FUNCTION Control (refNum: INTEGER; csCode: INTEGER; csParamPtr: Ptr) : OSErr;
FUNCTION Status (refNum: INTEGER; csCode: INTEGER; csParamPtr: Ptr) : OSErr;
FUNCTION KillIO (refNum: INTEGER) : OSErr;

Low-Level Routines

FUNCTION PBOpen (paramBlock: ParmBlkPtr; async: BOOLEAN) : OSErr;
  → 12 ioCompletion pointer
  → 16 ioResult word
  → 18 ioNamePtr pointer
  → 24 ioRefNum word
  → 27 ioPermssn byte

III-74 Device Manager
FUNCTION PBClose (paramBlock: ParmBlkPtr; async: BOOLEAN) : OSErr;
→ 12 ioCompletion pointer
← 16 ioResult word
→ 24 ioRefNum word

FUNCTION PBRead (paramBlock: ParmBlkPtr; async: BOOLEAN) : OSErr;
→ 12 ioCompletion pointer
← 16 ioResult word
→ 22 ioVRefNum word
→ 24 ioRefNum word
→ 32 ioBuffer pointer
→ 36 ioReqCount long word
← 40 ioActCount long word
← 44 ioPosMode word
← 46 ioPosOffset long word

FUNCTION PBWrite (paramBlock: ParmBlkPtr; async: BOOLEAN) : OSErr;
→ 12 ioCompletion pointer
← 16 ioResult word
→ 22 ioVRefNum word
→ 24 ioRefNum word
→ 32 ioBuffer pointer
→ 36 ioReqCount long word
← 40 ioActCount long word
← 44 ioPosMode word
← 46 ioPosOffset long word

FUNCTION PBControl (paramBlock: ParmBlkPtr; async: BOOLEAN) : OSErr;
→ 12 ioCompletion pointer
← 16 ioResult word
→ 22 ioVRefNum word
→ 24 ioRefNum word
→ 26 csCode word
→ 28 csParam record

FUNCTION PBStatus (paramBlock: ParmBlkPtr; async: BOOLEAN) : OSErr;
→ 12 ioCompletion pointer
← 16 ioResult word
→ 22 ioVRefNum word
→ 24 ioRefNum word
← 26 csCode word
← 28 csParam record

FUNCTION PBKillIO (paramBlock: ParmBlkPtr; async: BOOLEAN) : OSErr;
→ 12 ioCompletion pointer
← 16 ioResult word
← 24 ioRefNum word
Accessing a Driver's Device Control Entry

FUNCTION GetDCtlEntry (refNum: INTEGER) : DctlHandle; [Not in ROM]

Result Codes

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>abortErr</td>
<td>-27</td>
<td>I/O request aborted by KillIO</td>
</tr>
<tr>
<td>badUnitErr</td>
<td>-21</td>
<td>Driver reference number doesn't match unit table</td>
</tr>
<tr>
<td>controlErr</td>
<td>-17</td>
<td>Driver can't respond to this Control call</td>
</tr>
<tr>
<td>dlnstErr</td>
<td>-26</td>
<td>Couldn't find driver in resource file</td>
</tr>
<tr>
<td>dRemovErr</td>
<td>-25</td>
<td>Attempt to remove an open driver</td>
</tr>
<tr>
<td>noErr</td>
<td>0</td>
<td>No error</td>
</tr>
<tr>
<td>notOpenErr</td>
<td>-28</td>
<td>Driver isn't open</td>
</tr>
<tr>
<td>openErr</td>
<td>-23</td>
<td>Requested read/write permission doesn't match driver's open permission</td>
</tr>
<tr>
<td>readErr</td>
<td>-19</td>
<td>Driver can't respond to Read calls</td>
</tr>
<tr>
<td>statusErr</td>
<td>-18</td>
<td>Driver can't respond to this Status call</td>
</tr>
<tr>
<td>unitEmptyErr</td>
<td>-22</td>
<td>Driver reference number specifies NIL handle in unit table</td>
</tr>
<tr>
<td>writErr</td>
<td>-20</td>
<td>Driver can't respond to Write calls</td>
</tr>
</tbody>
</table>

Assembly-Language Information

Constants

; Flags in trap words

asyncTrpBit .EQU 10 ; set for an asynchronous call
noQueueBit .EQU 9  ; set for immediate execution

; Values for requesting read/write access

fsCurPerm   .EQU 0  ; whatever is currently allowed
fsRdPerm    .EQU 1  ; request to read only
fsWrPerm    .EQU 2  ; request to write only
fsRdWrPerm  .EQU 3  ; request to read and write

; Positioning modes

fsAtMark .EQU 0  ; at current position
fsFromStart .EQU 1 ; offset relative to beginning of medium
fsFromMark .EQU 3 ; offset relative to current position
rdVerify .EQU 64 ; add to above for read-verify
Summar y

; Driver flags

dReadEnable .EQU 0 ;set if driver can respond to Read calls
dWriteEnable .EQU 1 ;set if driver can respond to Write calls
dCtlEnable .EQU 2 ;set if driver can respond to Control calls
dStatEnable .EQU 3 ;set if driver can respond to Status calls
dNeedGoodBye .EQU 4 ;set if driver needs to be called before the
; application heap is reinitialized
dNeedTime .EQU 5 ;set if driver needs time for performing a
; periodic action
dNeedLock .EQU 6 ;set if driver will be locked in memory as
; soon as it's opened (always set for ROM
; drivers)

; Device control entry flags

dOpened .EQU 5 ;set if driver is open
dRABased .EQU 6 ;set if driver is RAM-based
drvrActive .EQU 7 ; set if driver is currently executing

; csCode values for driver control routine

accRun .EQU 65 ;take the periodic action, if any, for this
; driver
goodBye .EQU -1 ;heap will be reinitialized, clean up if
; necessary
killCode .EQU 1 ;handle the KillIO call

; Low-order byte of Device Manager traps

aRdCmd .EQU 2 ;Read call (trap $A002)
aWrCmd .EQU 3 ;Write call (trap $A003)

; Offsets from SCC base addresses

aData .EQU 6 ;channel A data in or out
actl .EQU 2 ;channel A control
bData .EQU 4 ;channel B data in or out
bCtl .EQU 0 ;channel B control

Standard Parameter Block Data Structure

qLink Pointer to next queue entry
qType Queue type (word)
ioTrap Routine trap (word)
ioCmdAddr Routine address
ioCompletion Address of completion routine
ioResult Result code (word)
ioVPNPtr Pointer to driver name (preceded by length byte)
ioVRefNum Volume reference number (word)
ioDrvNum Drive number (word)
Inside Macintosh

Control and Status Parameter Block Data Structure

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ioRefNum</td>
<td>Driver reference number (word)</td>
</tr>
<tr>
<td>csCode</td>
<td>Type of Control or Status call (word)</td>
</tr>
<tr>
<td>csParam</td>
<td>Parameters for Control or Status call (22 bytes)</td>
</tr>
</tbody>
</table>

I/O Parameter Block Data Structure

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ioRefNum</td>
<td>Driver reference number (word)</td>
</tr>
<tr>
<td>ioPermssn</td>
<td>Open permission (byte)</td>
</tr>
<tr>
<td>ioBuffer</td>
<td>Pointer to data buffer</td>
</tr>
<tr>
<td>ioReqCount</td>
<td>Requested number of bytes (long)</td>
</tr>
<tr>
<td>ioActCount</td>
<td>Actual number of bytes (long)</td>
</tr>
<tr>
<td>ioPosMode</td>
<td>Positioning mode (word)</td>
</tr>
<tr>
<td>ioPosOffset</td>
<td>Positioning offset (long)</td>
</tr>
</tbody>
</table>

Device Driver Data Structure

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>drvrFlags</td>
<td>Flags (word)</td>
</tr>
<tr>
<td>drvrDelay</td>
<td>Number of ticks between periodic actions (word)</td>
</tr>
<tr>
<td>drvrEMask</td>
<td>Desk accessory event mask (word)</td>
</tr>
<tr>
<td>drvrMenu</td>
<td>Menu ID of menu associated with driver (word)</td>
</tr>
<tr>
<td>drvrOpen</td>
<td>Offset to open routine (word)</td>
</tr>
<tr>
<td>drvrPrime</td>
<td>Offset to prime routine (word)</td>
</tr>
<tr>
<td>drvrCtl</td>
<td>Offset to control routine (word)</td>
</tr>
<tr>
<td>drvrStatus</td>
<td>Offset to status routine (word)</td>
</tr>
<tr>
<td>drvrClose</td>
<td>Offset to close routine (word)</td>
</tr>
<tr>
<td>drvrName</td>
<td>Driver name (preceded by length byte)</td>
</tr>
</tbody>
</table>

Device Control Entry Data Structure

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dCtlDriver</td>
<td>Pointer to ROM driver or handle to RAM driver</td>
</tr>
<tr>
<td>dCtlFlags</td>
<td>Flags (word)</td>
</tr>
<tr>
<td>dCtlQueue</td>
<td>Queue flags: low-order byte is driver's version number (word)</td>
</tr>
<tr>
<td>dCtlQHead</td>
<td>Pointer to first entry in driver's I/O queue</td>
</tr>
<tr>
<td>dCtlQTail</td>
<td>Pointer to last entry in driver's I/O queue</td>
</tr>
<tr>
<td>dCtlPosition</td>
<td>Byte position used by Read and Write calls (long)</td>
</tr>
<tr>
<td>dCtlStorage</td>
<td>Handle to RAM driver's private storage</td>
</tr>
<tr>
<td>dCtlRefNum</td>
<td>Driver's reference number (word)</td>
</tr>
<tr>
<td>dCtlWindow</td>
<td>Pointer to driver's window</td>
</tr>
<tr>
<td>dCtlDelay</td>
<td>Number of ticks between periodic actions (word)</td>
</tr>
<tr>
<td>dCtlEMask</td>
<td>Desk accessory event mask (word)</td>
</tr>
<tr>
<td>dCtlMenu</td>
<td>Menu ID of menu associated with driver (word)</td>
</tr>
</tbody>
</table>

Structure of Primary Interrupt Vector Table

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>autoInt1</td>
<td>Vector to level-1 interrupt handler</td>
</tr>
<tr>
<td>autoInt2</td>
<td>Vector to level-2 interrupt handler</td>
</tr>
</tbody>
</table>

III-78 Device Manager
autoInt3  Vector to level-3 interrupt handler
autoInt4  Vector to level-4 interrupt handler
autoInt5  Vector to level-5 interrupt handler
autoInt6  Vector to level-6 interrupt handler
autoInt7  Vector to level-7 interrupt handler

Macro Names

<table>
<thead>
<tr>
<th>Pascal name</th>
<th>Macro name</th>
</tr>
</thead>
<tbody>
<tr>
<td>PBRead</td>
<td>_Read</td>
</tr>
<tr>
<td>PBWrite</td>
<td>_Write</td>
</tr>
<tr>
<td>PBControl</td>
<td>_Control</td>
</tr>
<tr>
<td>PBStatus</td>
<td>_Status</td>
</tr>
<tr>
<td>PBKillIO</td>
<td>_KillIO</td>
</tr>
</tbody>
</table>

Routines for Writing Drivers

<table>
<thead>
<tr>
<th>Routine</th>
<th>Jump vector</th>
<th>On entry</th>
<th>On exit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fetch</td>
<td>JFetch</td>
<td>A1: ptr to device control entry</td>
<td>D0: character fetched; bit 15=1 if last character in buffer</td>
</tr>
<tr>
<td>Stash</td>
<td>JStash</td>
<td>A1: ptr to device control entry  D0: character to stash</td>
<td></td>
</tr>
<tr>
<td>IODone</td>
<td>JIODone</td>
<td>A1: ptr to device control entry  D0: result code (word)</td>
<td></td>
</tr>
</tbody>
</table>

Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UTableBase</td>
<td>Base address of unit table</td>
</tr>
<tr>
<td>JFetch</td>
<td>Jump vector for Fetch function</td>
</tr>
<tr>
<td>JStash</td>
<td>Jump vector for Stash function</td>
</tr>
<tr>
<td>JIODone</td>
<td>Jump vector for IODone function</td>
</tr>
<tr>
<td>Lvl1DT</td>
<td>Level-1 secondary interrupt vector table (32 bytes)</td>
</tr>
<tr>
<td>Lvl2DT</td>
<td>Level-2 secondary interrupt vector table (32 bytes)</td>
</tr>
<tr>
<td>VIA</td>
<td>VIA base address</td>
</tr>
<tr>
<td>ExtStsDT</td>
<td>External/status interrupt vector table (16 bytes)</td>
</tr>
<tr>
<td>SCCWr</td>
<td>SCC write base address</td>
</tr>
<tr>
<td>SCCRd</td>
<td>SCC read base address</td>
</tr>
</tbody>
</table>
DIALOG MANAGER

Constants

CONST { Item types }

ctrlItem = 4; {add to following four constants}
btnCtrl = 0; {standard button control}
chkCtrl = 1; {standard check box control}
rادCtrl = 2; {standard radio button control}
resCtrl = 3; {control defined in control template}
statText = 8; {static text}
ediTText = 16; {editable text (dialog only)}
iconItem = 32; {icon}
picItem = 64; {QuickDraw picture}
userItem = 0; {application-defined item (dialog only)}
itemDisable = 128; {add to any of above to disable}

{ Item numbers of OK and Cancel buttons }

ok = 1;
cancel = 2;

{ Resource IDs of alert icons }

stopIcon = 0;
noteIcon = 1;
cautionIcon = 2;

Data Types

TYPE DialogPtr = WindowPt.r;
DialogPeek = ^DialogRecord;

DialogRecord =
RECORD
window: WindowRecord; {dialog window}
items: Handle; {item list}
textH: TEHandle; {current editText item}
editField: INTEGER; {editText item number minus 1}
editOpen: INTEGER; {used internally}
aDefItem: INTEGER {default button item number}
END;

DialogTHndl = ^DialogTPtr;
DialogTPtr = ^DialogTemplate;
DialogTemplate =
  RECORD
    boundsRect: Rect; {becomes window's portRect}
    procID: INTEGER; {window definition ID}
    visible: BOOLEAN; {TRUE if visible}
    filler1: BOOLEAN; {not used}
    goAwayFlag: BOOLEAN; {TRUE if has go-away region}
    filler2: BOOLEAN; {not used}
    refCon: LONGINT; {window's reference value}
    itemsID: INTEGER; {resource ID of item list}
    title: Str255 [window's title]
  END;

AlertTHndl = ^AlertTPtr;
AlertTPtr = ^AlertTemplate;
AlertTemplate = RECORD
  boundsRect: Rect; {becomes window's portRect}
  itemsID: INTEGER; {resource ID of item list}
  stages: StageList (alert stage information)
END;

StageList = PACKED RECORD
  boldItm4: 0..1; {default button item number minus 1}
  boxDrwn4: BOOLEAN; {TRUE if alert box to be drawn}
  sound4: 0..3 {sound number}
  boldItm3: 0..1;
  boxDrwn3: BOOLEAN;
  sound3: 0..3
  boldItm2: 0..1;
  boxDrwn2: BOOLEAN;
  sound2: 0..3
  boldItm1: 0..1;
  boxDrwn1: BOOLEAN;
  sound1: 0..3
END;

Routines

Initialization

PROCEDURE InitDialogs (resumeProc: ProcPtr);
PROCEDURE ErrorSound (soundProc: ProcPtr);
PROCEDURE SetDAFont (fontNum: INTEGER); [Not in ROM]

Creating and Disposing of Dialogs

FUNCTION NewDialog (dStorage: Ptr; boundsRect: Rect; title: Str255;
  visible: BOOLEAN; procID: INTEGER; behind:
  WindowPtr; goAwayFlag: BOOLEAN; refCon: LONGINT;
  items: Handle) : DialogPtr;
FUNCTION GetNewDialog (dialogID: INTEGER; dStorage: Ptr; behind: WindowPtr: DialogPtr);
PROCEDURE CloseDialog (theDialog: DialogPtr);
PROCEDURE DisposeDialog (theDialog: DialogPtr);
PROCEDURE CouldDialog (dialogID: INTEGER);
PROCEDURE FreeDialog (dialogID: INTEGER);

Handling Dialog Events

PROCEDURE ModalDialog (filterProc: ProcPtr; VAR itemHit: INTEGER);
FUNCTION IsDialogEvent (theEvent: EventRecord) : BOOLEAN;
FUNCTION DialogSelect (theEvent: EventRecord; VAR theDialog: DialogPtr; VAR itemHit: INTEGER) : BOOLEAN;
PROCEDURE DlgCut (theDialog: DialogPtr); [Not in ROM]
PROCEDURE DlgCopy (theDialog: DialogPtr); [Not in ROM]
PROCEDURE DlgPaste (theDialog: DialogPtr); [Not in ROM]
PROCEDURE DlgDelete (theDialog: DialogPtr); [Not in ROM]
PROCEDURE DrawDialog (theDialog: DialogPtr);

Invoking Alerts

FUNCTION Alert (alertID: INTEGER; filterProc: ProcPtr) : INTEGER;
FUNCTION StopAlert (alertID: INTEGER; filterProc: ProcPtr) : INTEGER;
FUNCTION NoteAlert (alertID: INTEGER; filterProc: ProcPtr) : INTEGER;
FUNCTION CautionAlert (alertID: INTEGER; filterProc: ProcPtr) : INTEGER;
PROCEDURE CouldAlert (alertID: INTEGER);
PROCEDURE FreeAlert (alertID: INTEGER);

Manipulating Items in Dialogs and Alerts

PROCEDURE ParamText (param0,param1,param2,param3: Str255);
PROCEDURE GetDItem (theDialog: DialogPtr; itemNo: INTEGER; VAR itemType: INTEGER; VAR item: Handle; VAR box: Rect);
PROCEDURE SetDItem (theDialog: DialogPtr; itemNo: INTEGER; item: Handle; box: Rect);
PROCEDURE GetIText (item: Handle; text: Str255);
PROCEDURE SetIText (item: Handle; text: Str255);
PROCEDURE SelIText (theDialog: DialogPtr; itemNo: INTEGER; startSel,endSel: INTEGER);
FUNCTION GetAlertStage : INTEGER; [Not in ROM]
PROCEDURE ResetAlertStage; [Not in ROM]

UserItem Procedure

PROCEDURE MyItem (theWindow: WindowPtr; itemNo: INTEGER);
Sound Procedure

PROCEDURE MySound (soundNo: INTEGER);

FilterProc Function for Modal Dialogs and Alerts

FUNCTION MyFilter (theDialog: DialogPtr; VAR theEvent: EventRecord; VAR itemHit: INTEGER) : BOOLEAN;

Assembly-Language Information

Constants

; Item types
ctrlItem .EQU 4 ;add to following four constants
btnCtrl .EQU 0 ;standard button control
chkCtrl .EQU 1 ;standard check box control
radCtrl .EQU 2 ;standard radio button control
resCtrl .EQU 3 ;control defined in control template
statText .EQU 8 ;static text
editText .EQU 16 ;editable text (dialog only)
iconItem .EQU 32 ;icon
picItem .EQU 64 ;QuickDraw picture
userItem .EQU 0 ;application-defined item (dialog only)
itemDisable .EQU 128 ;add to any of above to disable

; Item numbers of OK and Cancel buttons
okButton .EQU 1
cancelButton .EQU 2

; Resource IDs of alert icons
stopIcon .EQU 0
noteIcon .EQU 1
cautIcon .EQU 2

; Masks for stages word in alert template
volBits .EQU 3 ;sound number
alBit .EQU 4 ;whether to draw box
okDismissal .EQU 8 ;item number of default button minus 1
Inside Macintosh

Dialog Record Data Structure

dWindow : Dialog window
items : Handle to dialog's item list
tHandle : Handle to current editField item
editField : Item number of editField item minus 1 (word)
aDefItem : Item number of default button (word)
dWindLen : Size in bytes of dialog record

Dialog Template Data Structure

dBounds : Rectangle that becomes portRect of dialog window's grafPort (8 bytes)
dWindProc : Window definition ID (word)
dVisible : Nonzero if dialog window is visible (word)
dGoAway : Nonzero if dialog window has a go-away region (word)
dRefCon : Dialog window's reference value (long)
dItems : Resource ID of dialog's item list (word)
dTitle : Dialog window's title (preceded by length byte)

Alert Template Data Structure

aBounds : Rectangle that becomes portRect of alert window's grafPort (8 bytes)
aItems : Resource ID of alert's item list (word)
aStages : Stages word; information for alert stages

Item List Data Structure

dlgMaxIndex : Number of items minus 1 (word)
itmHndl : Handle or procedure pointer for this item
itmRect : Display rectangle for this item (8 bytes)
itmType : Item type for this item (byte)
itmData : Length byte followed by data for this item (data must be even number of bytes)

Variables

ResumeProc : Address of resume procedure
DAStrings : Handles to ParamText strings (16 bytes)
DABeeper : Address of current sound procedure
DlgFont : Font number for dialogs and alerts (word)
ACount : Stage number (0 through 3) of last alert (word)
ANumber : Resource ID of last alert (word)

III-84 Dialog Manager
DISK DRIVER

Constants

CONST { Positioning modes }

fsAtMark = 0;  {at current sector}
fsFromStart = 1;  {relative to first sector}
fsFromMark = 3;  {relative to current sector}
rddVerify = 64;  {add to above for read-verify}

Data Types

TYPE DrvSts = RECORD
  track: INTEGER;  {current track}
  writeProt: SignedByte;  {bit 7=1 if volume is locked}
  diskInPlace: SignedByte;  {disk in place}
  installed: SignedByte;  {drive installed}
  sides: SignedByte;  {bit 7=0 if single-sided drive}
  qLink: QElemPtr;  {next queue entry}
  qType: INTEGER;  {reserved for future use}
  dQDrive: INTEGER;  {drive number}
  dQRefNum: INTEGER;  {driver reference number}
  dQFSID: INTEGER;  {file-system identifier}
  twoSideFmt: SignedByte;  {-1 if two-sided disk}
  needsFlush: SignedByte;  {reserved for future use}
  diskErrs: INTEGER  {error count}
END;

Routines [Not in ROM]

FUNCTION DiskEject (drvNum: INTEGER) : OSErr;
FUNCTION SetTagBuffer (buffPtr: Ptr) : OSErr;
FUNCTION DriveStatus (drvNum: INTEGER; VAR status : DrvSts) : OSErr;

Result Codes

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>noErr</td>
<td>0</td>
<td>No error</td>
</tr>
<tr>
<td>nsDrvErr</td>
<td>-56</td>
<td>No such drive</td>
</tr>
<tr>
<td>paramErr</td>
<td>-50</td>
<td>Bad positioning information</td>
</tr>
<tr>
<td>wPrErr</td>
<td>-44</td>
<td>Volume is locked by a hardware setting</td>
</tr>
<tr>
<td>Name</td>
<td>Value</td>
<td>Meaning</td>
</tr>
<tr>
<td>-------------------</td>
<td>-------</td>
<td>----------------------------------------------------------</td>
</tr>
<tr>
<td>firstDskErr</td>
<td>-84</td>
<td>First of the range of low-level disk errors</td>
</tr>
<tr>
<td>sectNFErr</td>
<td>-81</td>
<td>Can't find sector</td>
</tr>
<tr>
<td>seekErr</td>
<td>-80</td>
<td>Drive error</td>
</tr>
<tr>
<td>spdAdjErr</td>
<td>-79</td>
<td>Can't correctly adjust disk speed</td>
</tr>
<tr>
<td>twoSideErr</td>
<td>-78</td>
<td>Tried to read side 2 of a disk in a single-sided drive</td>
</tr>
<tr>
<td>initWMErr</td>
<td>-77</td>
<td>Can't initialize disk controller chip</td>
</tr>
<tr>
<td>tk0BadErr</td>
<td>-76</td>
<td>Can't find track 0</td>
</tr>
<tr>
<td>cantStepErr</td>
<td>-75</td>
<td>Drive error</td>
</tr>
<tr>
<td>wrUnderrun</td>
<td>-74</td>
<td>Write underrun occurred</td>
</tr>
<tr>
<td>badDBtSlp</td>
<td>-73</td>
<td>Bad data mark</td>
</tr>
<tr>
<td>badDCKsum</td>
<td>-72</td>
<td>Bad data mark</td>
</tr>
<tr>
<td>noDtaMkErr</td>
<td>-71</td>
<td>Can't find data mark</td>
</tr>
<tr>
<td>badBtSlpErr</td>
<td>-70</td>
<td>Bad address mark</td>
</tr>
<tr>
<td>badCksmErr</td>
<td>-69</td>
<td>Bad address mark</td>
</tr>
<tr>
<td>dataVerErr</td>
<td>-68</td>
<td>Read-verify failed</td>
</tr>
<tr>
<td>noAdrMkErr</td>
<td>-67</td>
<td>Can't find an address mark</td>
</tr>
<tr>
<td>noNybErr</td>
<td>-66</td>
<td>Disk is probably blank</td>
</tr>
<tr>
<td>offLinErr</td>
<td>-65</td>
<td>No disk in drive</td>
</tr>
<tr>
<td>noDriveErr</td>
<td>-64</td>
<td>Drive isn't connected</td>
</tr>
<tr>
<td>lastDskErr</td>
<td>-64</td>
<td>Last of the range of low-level disk errors</td>
</tr>
</tbody>
</table>

**Assembly-Language Information**

### Constants

; Positioning modes

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fsAtMark</td>
<td>.EQU  0</td>
<td>at current sector</td>
</tr>
<tr>
<td>fsFromStart</td>
<td>.EQU 1</td>
<td>relative to first sector</td>
</tr>
<tr>
<td>fsFromMark</td>
<td>.EQU 3</td>
<td>relative to current sector</td>
</tr>
<tr>
<td>rdVerify</td>
<td>.EQU 64</td>
<td>add to above for read-verify</td>
</tr>
</tbody>
</table>

; csCode values for Control/Status calls

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ejectCode</td>
<td>.EQU 7</td>
<td>Control call, DiskEject</td>
</tr>
<tr>
<td>tgBuffCode</td>
<td>.EQU 8</td>
<td>Control call, SetTagBuffer</td>
</tr>
<tr>
<td>drvStsCode</td>
<td>.EQU 8</td>
<td>Status call, DriveStatus</td>
</tr>
</tbody>
</table>

### Structure of Status Information

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dsTrack</td>
<td>Current track (word)</td>
</tr>
<tr>
<td>dsWriteProt</td>
<td>Bit 7=1 if volume is locked (byte)</td>
</tr>
<tr>
<td>dsDiskInPlace</td>
<td>Disk in place (byte)</td>
</tr>
<tr>
<td>dsInstalled</td>
<td>Drive installed (byte)</td>
</tr>
<tr>
<td>dsSides</td>
<td>Bit 7=0 if single-sided drive (byte)</td>
</tr>
<tr>
<td>dsQLink</td>
<td>Pointer to next queue entry</td>
</tr>
<tr>
<td>dsDQDrive</td>
<td>Drive number (word)</td>
</tr>
</tbody>
</table>

*III-86 Disk Driver*
dsDQRefNum  Driver reference number (word)
dsDQFSID     File-system identifier (word)
dsTwoSideFmt –1 if two-sided disk (byte)
dsDiskErrs   Error count (word)

**Equivalent Device Manager Calls**

<table>
<thead>
<tr>
<th>Pascal routine</th>
<th>Call</th>
</tr>
</thead>
<tbody>
<tr>
<td>DiskEject</td>
<td>Control with csCode=ejectCode</td>
</tr>
<tr>
<td>SetTagBuffer</td>
<td>Control with csCode=tgBuffCode</td>
</tr>
<tr>
<td>DriveStatus</td>
<td>Status with csCode=drvStsCode, status returned in csParam through csParam+21</td>
</tr>
</tbody>
</table>

**Variables**

- BufTgFNum: File tags buffer: file number (long)
- BufTgFFlag: File tags buffer: flags (word: bit 1=1 if resource fork)
- BufTgFBkNum: File tags buffer: logical block number (word)
- BufTgDate: File tags buffer: date and time of last modification (long)
DISK INITIALIZATION PACKAGE

Routines

PROCEDURE DILoad;
PROCEDURE DIUnload;
FUNCTION DIBadMount (where: Point; evtMessage: LONGINT) : INTEGER;
FUNCTION DIFormat (drvNum: INTEGER) : OSErr;
FUNCTION DIFverify (drvNum: INTEGER) : OSErr;
FUNCTION DIZero (drvNum: INTEGER; volName: Str255) : OSErr;

Result Codes

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>badMDBErr</td>
<td>-60</td>
<td>Bad master directory block</td>
</tr>
<tr>
<td>extFSErr</td>
<td>-58</td>
<td>External file system</td>
</tr>
<tr>
<td>firstDskErr</td>
<td>-84</td>
<td>First of the range of low-level disk errors</td>
</tr>
<tr>
<td>ioErr</td>
<td>-36</td>
<td>I/O error</td>
</tr>
<tr>
<td>lastDskErr</td>
<td>-64</td>
<td>Last of the range of low-level disk errors</td>
</tr>
<tr>
<td>memFullErr</td>
<td>-108</td>
<td>Not enough room in heap zone</td>
</tr>
<tr>
<td>noErr</td>
<td>0</td>
<td>No error</td>
</tr>
<tr>
<td>noMacDskErr</td>
<td>-57</td>
<td>Not a Macintosh disk</td>
</tr>
<tr>
<td>nsDrvErr</td>
<td>-56</td>
<td>No such drive</td>
</tr>
<tr>
<td>paramErr</td>
<td>-50</td>
<td>Bad drive number</td>
</tr>
<tr>
<td>volOnLinErr</td>
<td>-55</td>
<td>Volume already on-line</td>
</tr>
</tbody>
</table>

Assembly-Language Information

Constants

; Routine selectors

diBadMount .EQU 0
dILoad .EQU 2
dIUUnload .EQU 4
dIFormat .EQU 6
dIVerify .EQU 8
dIZero .EQU 10

III-88 Disk Initialization Package
Trap Macro Name

_Pack2
EVENT MANAGER, OPERATING SYSTEM

Constants

CONST { Event codes }

nullEvent = 0;   {null}
mouseDown = 1;   {mouse-down}
mouseUp = 2;     {mouse-up}
keyDown = 3;     {key-down}
keyUp = 4;       {key-up}
autoKey = 5;     {auto-key}
updateEvt = 6;   {update; Toolbox only}
diskEvt = 7;     {disk-inserted}
activateEvt = 8; {activate; Toolbox only}
networkEvt = 10; {network}
driverEvt = 11;  {device driver}
app1Evt = 12;    {application-defined}
app2Evt = 13;    {application-defined}
app3Evt = 14;    {application-defined}
app4Evt = 15;    {application-defined}

{ Masks for keyboard event message }

charCodeMask = $000000FF; {character code}
keyCodeMask = $0000FF00;  {key code}

{ Masks for forming event mask }

mDownMask = 2;   {mouse-down}
mUpMask = 4;     {mouse-up}
keyDownMask = 8; {key-down}
keyUpMask = 16;  {key-up}
autoKeyMask = 32; {auto-key}
updateMask = 64; {update}
diskMask = 128;  {disk-inserted}
activateMask = 256; {activate}
networkMask = 1024; {network}
driverMask = 2048; {device driver}
app1Mask = 4096; {application-defined}
app2Mask = 8192; {application-defined}
app3Mask = 16384; {application-defined}
app4Mask = -32768; {application-defined}
everyEvent = -1; {all event types}
{ Modifier flags in event record }

activeFlag = 1; {set if window being activated}
btnState = 128; {set if mouse button up}
cmdKey = 256; {set if Command key down}
shiftKey = 512; {set if Shift key down}
alphaLock = 1024; {set if Caps Lock key down}
optionKey = 2048; {set if Option key down}

{ Result codes returned by PostEvent }
noErr = 0; {no error (event posted)}
evtNotEnb = 1; {event type not designated in system event mask}

Data Types

TYPE EventRecord = RECORD
  what: INTEGER; {event code}
  message: LONGINT; {event message}
  when: LONGINT; {ticks since startup}
  where: Point; {mouse location}
  modifiers: INTEGER {modifier flags}
END;

EvQEl = RECORD
  qLink: QElemPtr; {next queue entry}
  qType: INTEGER; {queue type}
  evtQWhat: INTEGER {event code}
  evtQMessage: LONGINT; {event message}
  evtQWhen: LONGINT; {ticks since startup}
  evtQWhere: Point; {mouse location}
  evtQModifiers: INTEGER {modifier flags}
END;

Routines

Posting and Removing Events

FUNCTION PostEvent (eventCode: INTEGER; eventMsg: LONGINT) : OSErr;
PROCEDURE FlushEvents (eventMask, stopMask: INTEGER);

Accessing Events

FUNCTION GetOSEvent (eventMask: INTEGER; VAR theEvent: EventRecord) : BOOLEAN;
FUNCTION OSEventAvail (eventMask: INTEGER; VAR theEvent: EventRecord) : BOOLEAN;
Setting the System Event Mask

PROCEDURE SetEventMask (theMask: INTEGER); [Not in ROM]

Directly Accessing the Event Queue

FUNCTION GetEvQHdr : QHdrPtr; [Not in ROM]

Assembly-Language Information

Constants

; Event codes

nullEvt .EQU 0 ; null
mButDwnEvt .EQU 1 ; mouse-down
mButUpEvt .EQU 2 ; mouse-up
keyDwnEvt .EQU 3 ; key-down
keyUpEvt .EQU 4 ; key-up
autoKeyEvt .EQU 5 ; auto-key
updatEvt .EQU 6 ; update; Toolbox only
diskInsertEvt .EQU 7 ; disk-inserted
activateEvt .EQU 8 ; activate; Toolbox only
networkEvt .EQU 10 ; network
ioDrvrEvt .EQU 11 ; device driver
app1Evt .EQU 12 ; application-defined
app2Evt .EQU 13 ; application-defined
app3Evt .EQU 14 ; application-defined
app4Evt .EQU 15 ; application-defined

; Modifier flags in event record

activeFlag .EQU 0 ; set if window being activated
btnState .EQU 2 ; set if mouse button up
cmdKey .EQU 3 ; set if Command key down
shiftKey .EQU 4 ; set if Shift key down
alphaLock .EQU 5 ; set if Caps Lock key down
optionKey .EQU 6 ; set if Option key down

; Result codes returned by PostEvent

noErr .EQU 0 ; no error (event posted)
evtNotEnb .EQU 1 ; event type not designated in system ; event mask

III-92 Event Manager, Operating System
Event Record Data Structure

evtNum  Event code (word)
evtMessage Event message (long)
evtTicks  Ticks since startup (long)
evtMouse  Mouse location (point; long)
evtMeta  State of modifier keys (byte)
evtMBut  State of mouse button (byte)
evtBlkSize  Size in bytes of event record

Event Queue Entry Data Structure

qLink  Pointer to next queue entry
qType  Queue type (word)
evtQWhat  Event code (word)
evtQMessage Event message (long)
evtQWhen  Ticks since startup (long)
evtQWhere  Mouse location (point; long)
evtQMeta  State of modifier keys (byte)
evtQMBut  State of mouse button (byte)
evtQBBlkSize  Size in bytes of event queue entry

Routines

- Trap macro
  _PostEvent  On entry
  A0:  eventCode (word)
  D0:  eventMsg (long)
  D0:  low word: eventMask
  high word: stopMask
  A0:  ptr to event record
  theEvent
  D0:  result code (word)
  D0:  0 or event code (word)
  D0:  0 if non-null event,
  -1 if null event (byte)

- _FlushEvents
  D0:  0 or event code (word)

- _GetOSEvent
  A0:  ptr to event record
  theEvent
  D0:  eventMask (word)

- _OSEventAvail
  D0:  eventMask (word)

Variables

SysEvtMask  System event mask (word)
EventQueue  Event queue header (10 bytes)
EVENT MANAGER, TOOLBOX

Constants

CONST { Event codes }

nullEvent = 0; {null}
mouseDown = 1; {mouse down}
mouseUp = 2; {mouse up}
keyDown = 3; {key-down}
keyUp = 4; {key-up}
autoKey = 5; {auto-key}
updateEvt = 6; {update}
diskEvt = 7; {disk-inserted}
activateEvt = 8; {activate}
networkEvt = 10; {network}
driverEvt = 11; {device driver}
app1Evt = 12; {application-defined}
app2Evt = 13; {application-defined}
app3Evt = 14; {application-defined}
app4Evt = 15; {application-defined}

{ Masks for keyboard event message }

charCodeMask = $000000FF; {character code}
keyCodeMask = $0000FF00; {key code}

{ Masks for forming event mask }

mDownMask = 2; {mouse down}
mUpMask = 4; {mouse up}
keyDownMask = 8; {key-down}
keyUpMask = 16; {key-up}
autoKeyMask = 32; {auto-key}
updateMask = 64; {update}
diskMask = 128; {disk-inserted}
activMask = 256; {activate}
networkMask = 1024; {network}
driverMask = 2048; {device driver}
applMask = 4096; {application-defined}
app2Mask = 8192; {application-defined}
app3Mask = 16384; {application-defined}
app4Mask = -32768; {application-defined}
everyEvent = -1; {all event types}
{ Modifier flags in event record }

activeFlag = 1; {set if window being activated}
btnState = 128; {set if mouse button up}
cmdKey = 256; {set if Command key down}
shiftKey = 512; {set if Shift key down}
alphaLock = 1024; {set if Caps Lock key down}
optionKey = 2048; {set if Option key down}

Data Types

TYPE EventRecord = RECORD
  what: INTEGER; \{event code\}
  message: LONGINT; \{event message\}
  when: LONGINT; \{ticks since startup\}
  where: Point; \{mouse location\}
  modifiers: INTEGER \{modifier flags\}
END;

KeyMap = PACKED ARRAY[0..127] OF BOOLEAN;

Routines

Accessing Events

FUNCTION GetNextEvent (eventMask: INTEGER; VAR theEvent: EventRecord) : BOOLEAN;
FUNCTION EventAvail (eventMask: INTEGER; VAR theEvent: EventRecord) : BOOLEAN;

Reading the Mouse

PROCEDURE GetMouse (VAR mouseLoc: Point);
FUNCTION Button : BOOLEAN;
FUNCTION StillDown : BOOLEAN;
FUNCTION WaitMouseUp : BOOLEAN;

Reading the Keyboard and Keypad

PROCEDURE GetKeys (VAR theKeys: KeyMap);

Miscellaneous Routines

FUNCTIONTickCount : LONGINT;
FUNCTION GetDb1Time : LONGINT; \{Not in ROM\}
FUNCTION GetCaretTime : LONGINT; \{Not in ROM\}
Event Message in Event Record

<table>
<thead>
<tr>
<th>Event type</th>
<th>Event message</th>
</tr>
</thead>
<tbody>
<tr>
<td>Keyboard</td>
<td>Character code and key code in low-order word</td>
</tr>
<tr>
<td>Activate, update</td>
<td>Pointer to window</td>
</tr>
<tr>
<td>Disk-inserted</td>
<td>Drive number in low-order word, File Manager result code in high-order word</td>
</tr>
<tr>
<td>Mouse-down, mouse-up, null</td>
<td>Undefined</td>
</tr>
<tr>
<td>Network</td>
<td>Handle to parameter block</td>
</tr>
<tr>
<td>Device driver</td>
<td>See chapter describing driver</td>
</tr>
<tr>
<td>Application-defined</td>
<td>Whatever you wish</td>
</tr>
</tbody>
</table>

Assembly-Language Information

Constants

; Event codes

nullEvt EQU 0 ;null
mButDwnEvt EQU 1 ;mouse-down
mButUpEvt EQU 2 ;mouse-up
keyDwnEvt EQU 3 ;key-down
keyUpEvt EQU 4 ;key-up
autoKeyEvt EQU 5 ;auto-key
updAtEvt EQU 6 ;update
disklnsertEvt EQU 7 ;disk-inserted
activateEvt EQU 8 ;activate
networkEvt EQU 10 ;network
ioDrvrEvt EQU 11 ;device driver
applEvt EQU 12 ;application-defined
app2Evt EQU 13 ;application-defined
app3Evt EQU 14 ;application-defined
app4Evt EQU 15 ;application-defined

; Modifier flags in event record

activeFlag EQU 0 ;set if window being activated
btnState EQU 2 ;set if mouse button up
cmdKey EQU 3 ;set if Command key down
shiftKey EQU 4 ;set if Shift key down
alphaLock EQU 5 ;set if Caps Lock key down
optionKey EQU 6 ;set if Option key down
; Journaling mechanism Control call

jPlayCtl  .EQU 16 ;journal in playback mode
jRecordCtl .EQU 17 ;journal in recording mode
jcTickCount .EQU 0 ;journal code for TickCount
jcGetMouse .EQU 1 ;journal code for GetMouse
jcMouseButton .EQU 2 ;journal code for Button
jcGetKeys .EQU 3 ;journal code for GetKeys
jcEvent . EQU 4 ;journal code for GetNextEvent and EventAvail

Event Record Data Structure

evtNum Event code (word)
evtMessage Event message (long)
evtTicks Ticks since startup (long)
evtMouse Mouse location (point; long)
evtMeta State of modifier keys (byte)
evtMBut State of mouse button (byte)
evtBlkSize Size in bytes of event record

Variables

KeyThresh Auto-key threshold (word)
KeyRepThresh Auto-key rate (word)
WindowList 0 if using events but not windows (long)
ScrDmpEnb 0 if GetNextEvent shouldn't process Command-Shift-number combinations (byte)
Ticks Current number of ticks since system startup (long)
DoubleTime Double-click interval in ticks (long)
CaretTime Caret-blink interval in ticks (long)
JournalRef Reference number of journaling device driver (word)
JournalFlag Journaling mode (word)
FILE MANAGER

Constants

CONST { Flags in file information used by the Finder }

fHasBundle = 8192; {set if file has a bundle}
fInvisible = 16384; {set if file's icon is invisible}
fTrash = -3; {file is in Trash window}
fDesktop = -2; {file is on desktop}
fDisk = 0; {file is in disk window}

{ Values for requesting read/write access }

fsCurPerm = 0; {whatever is currently allowed}
fsRdPerm = 1; {request to read only}
fsWrPerm = 2; {request to write only}
fsRdWrPerm = 3; {request to read and write}

{ Positioning modes }

fsAtMark = 0; {at current mark}
fsFromStart = 1; {offset relative to beginning of file}
fsFromLEOF = 2; {offset relative to logical end-of-file}
fsFromMark = 3; {offset relative to current mark}
rdVerify = 64; {add to above for read-verify}

Data Types

TYPE FInfo = RECORD
    fdType: OSType; {file type}
    fdCreator: OSType; {file's creator}
    fdFlags: INTEGER; {flags}
    fdLocation: Point; {file's location}
    fdFldr: INTEGER {file's window}
END;

ParamBlkType = (ioParam, fileParam, volumeParam, ctrlParam);

ParmBlkPtr = ^ParamBlockRec;
ParamBlockRec = RECORD
    qLink: QElemPtr; {next queue entry}
    qType: INTEGER; {queue type}
    ioTrap: INTEGER; {routine trap}
    ioCmdAddr: Ptr; {routine address}
    ioCompletion: ProcPtr; {completion routine}
    ioResult: OSErr; {result code}
    ioNamePtr: StringPtr; {volume or file name}
    ioVRefNum: INTEGER; {volume reference or drive number}
CASE ParamBlkType OF
  ioParam:
    (ioRefNum: INTEGER; {path reference number}
    ioVersNum: SignedByte; {version number}
    ioPermssn: SignedByte; {read/write permission}
    ioMisc: Ptr; {miscellaneous}
    ioBuffer: Ptr; {data buffer}
    ioReqCount: LONGINT; {requested number of bytes}
    ioActCount: LONGINT; {actual number of bytes}
    ioPosMode: INTEGER; {positioning mode and newline character}
    ioPosOffset: LONGINT; {positioning offset}
  fileParam:
    (ioFRefNum: INTEGER; {path reference number}
    ioFVersNum: SignedByte; {version number}
    filler1: SignedByte; {not used}
    ioFDirIndex: INTEGER; {sequence number of file}
    ioFlAttrib: SignedByte; {file attributes}
    ioFlVersNum: SignedByte {version number}
    ioFlFndrInfo: Finfo; {information used by the Finder}
    ioFlNum: LONGINT; {file number}
    ioFlStBlk: INTEGER; {first allocation block of data fork}
    ioFlLgLen: LONGINT; {logical end-of-file of data fork}
    ioFlPyLen: LONGINT; {physical end-of-file of data fork}
    ioFlRStBlk: INTEGER; {first allocation block of resource }
    ioFlRLgLen: LONGINT; {logical end-of-file of resource fork}
    ioFlRPyLen: LONGINT; {physical end-of-file of resource }
    ioFlCrDat: LONGINT; {date and time of creation}
    ioFlMdDat: LONGINT); {date and time of last modification}
  volumeParam:
    (filler2: LONGINT; {not used}
    ioVolIndex: INTEGER; {volume index}
    ioVCRDate: LONGINT; {date and time of initialization}
    ioVLSbkUp: LONGINT; {date and time of last backup}
    ioVAttrb: INTEGER; {bit 15=1 if volume locked}
    ioVNmFls: INTEGER; {number of files in directory}
    ioVDirSt: INTEGER; {first block of directory}
    ioVBlln: INTEGER; {length of directory in blocks}
    ioVNmAlBlks: INTEGER; {number of allocation blocks}
    ioVALBlksiz: LONGINT; {size of allocation blocks}
    ioVCplSiz: LONGINT; {number of bytes to allocate}
    ioAL.Blks: INTEGER; {first allocation block in block map}
    ioVNxtFNum: LONGINT; {next unused file number}
    ioVFrBlk: INTEGER); {number of unused allocation blocks}
  cntrlParam:
    . . . {used by Device Manager}
END;
VCB = RECORD
  qLink: QElemPtr;  {next queue entry}
  qType: INTEGER;  {queue type}
  vcbFlags: INTEGER;  {bit 15=1 if dirty}
  vcbSigWord: INTEGER;  {always $D2D7}
  vcbCrDate: LONGINT;  {date and time of initialization}
  vcbLsBkUp: LONGINT;  {date and time of last backup}
  vcbAttrb: INTEGER;  {volume attributes}
  vcbNmFls: INTEGER;  {number of files in directory}
  vcbDirSt: INTEGER;  {first block of directory}
  vcbBlLn: INTEGER;  {length of directory in blocks}
  vcbNmBlks: INTEGER;  {number of allocation blocks}
  vcbAlBlkSiz: LONGINT;  {size of allocation blocks}
  vcbClpSiz: LONGINT;  {number of bytes to allocate}
  vcbAlBlkSt: INTEGER;  {first allocation block in block map}
  vcbNxtFNum: INTEGER;  {next unused file number}
  vcbFreeBks: INTEGER;  {number of unused allocation blocks}
  vcbVN: STRING[27];  {volume name}
  vcbDrvNum INTEGER;  {drive number}
  vcbDRefNum: INTEGER;  {driver reference number}
  vcbFSID: INTEGER;  {file-system identifier}
  vcbVRefNum: INTEGER;  {volume reference number}
  vcbMaDr: Ptr;  {pointer to block map}
  vcbBufAdr: Ptr;  {pointer to volume buffer}
  vcbMLen: INTEGER;  {number of bytes in block map}
  vcbDirIndex: INTEGER;  {used internally}
  vcbDirBlk: INTEGER;  {used internally}
END;

DrvQEl = RECORD
  qLink: QElemPtr;  {next queue entry}
  qType: INTEGER;  {queue type}
  dQDrive: INTEGER;  {drive number}
  dQRefNum: INTEGER;  {driver reference number}
  dFSID: INTEGER;  {file-system identifier}
  dDrvSize: INTEGER;  {number of logical blocks}
END;

High-Level Routines  [Not in ROM]

Accessing Volumes

FUNCTION GetVInfo  (drvNum: INTEGER; volName: StringPtr; VAR vRefNum: INTEGER; VAR freeBytes: LONGINT) : OSErr;
FUNCTION GetVRefNum  (pathRefNum: INTEGER; VAR vRefNum: INTEGER) : OSErr;
FUNCTION GetVol  (volName: StringPtr; VAR vRefNum: INTEGER) : OSErr;
FUNCTION SetVol  (volName: StringPtr; vRefNum: INTEGER) : OSErr;
FUNCTION FlushVol  (volName: StringPtr; vRefNum: INTEGER) : OSErr;
FUNCTION UnmountVol  (volName: StringPtr; vRefNum: INTEGER) : OSErr;
FUNCTION Eject  (volName: StringPtr; vRefNum: INTEGER) : OSErr;

III-100  File Manager
**Accessing Files**

FUNCTION Create (fileName: Str255; vRefNum: INTEGER; creator: OSTYPE; fileType: OSTYPE) : OSErr;

FUNCTION FSOpen (fileName: Str255; vRefNum: INTEGER; VAR refNum: INTEGER) : OSErr;

FUNCTION OpenRF (fileName: Str255; vRefNum: INTEGER; refNum: INTEGER) : OSErr;

FUNCTION FSRead (refNum: INTEGER; VAR count: LONGINT; buffPtr: Ptr) : OSErr;

FUNCTION FSWrite (refNum: INTEGER; VAR count: LONGINT; buffPtr: Ptr) : OSErr;

FUNCTION GetFPos (refNum: INTEGER; VAR filePos: LONGINT) : OSErr;

FUNCTION SetFPos (refNum: INTEGER; posMode: INTEGER; posOff: LONGINT) : OSErr;

FUNCTION GetEOF (refNum: INTEGER; VAR logEOF: LONGINT) : OSErr;

FUNCTION SetEOF (refNum: INTEGER; logEOF: LONGINT) : OSErr;

FUNCTION Allocate (refNum: INTEGER; VAR count: LONGINT) : OSErr;

FUNCTION FSClose (refNum: INTEGER) : OSErr;

**Changing Information About Files**

FUNCTION GetFInfo (fileName: Str255; vRefNum: INTEGER; VAR fndrInfo: FInfo) : OSErr;

FUNCTION SetFInfo (fileName: Str255; vRefNum: INTEGER; fndrInfo: FInfo) : OSErr;

FUNCTION SetFLock (fileName: Str255; vRefNum: INTEGER) : OSErr;

FUNCTION RstFLock (fileName: Str255; vRefNum: INTEGER) : OSErr;

FUNCTION Rename (oldName: Str255; vRefNum: INTEGER; newName: Str255) : OSErr;

FUNCTION FSDelete (fileName: Str255; vRefNum: INTEGER) : OSErr;

**Low-Level Routines**

**Initializing the File I/O Queue**

PROCEDURE FInitQueue;

**Accessing Volumes**

FUNCTION PBMountVol (paramBlock: ParmBlkPtr) : OSErr;

← 16 ioResult word

← 22 ioVRefNum word
FUNCTION PBGetVInfo (paramBlock: ParmBlkPtr; async: BOOLEAN) : OSErr;

| 12 | ioCompletion       | pointer |
| 16 | ioResult           | word    |
| 18 | ioNamePtr          | pointer |
| 22 | ioVRefNum          | word    |
| 28 | ioVolIndex         | word    |
| 30 | ioVDate            | long word |
| 34 | ioVLSbkUp          | long word |
| 38 | ioVAtb             | word    |
| 40 | ioVNMFLs           | word    |
| 42 | ioVDirlSt          | word    |
| 44 | ioVBILn            | word    |
| 46 | ioVNmlblks         | word    |
| 48 | ioVAlblksSiz       | long word |
| 52 | ioVCipSiz          | long word |
| 56 | ioVBlSt            | word    |
| 58 | ioVNXfFNum         | long word |
| 62 | ioVFrstBlk         | word    |

FUNCTION PBGetVol (paramBlock: ParmBlkPtr; async: BOOLEAN) : OSErr;

| 12 | ioCompletion       | pointer |
| 16 | ioResult           | word    |
| 18 | ioNamePtr          | pointer |
| 22 | ioVRefNum          | word    |

FUNCTION PBSetVol (paramBlock: ParmBlkPtr; async: BOOLEAN) : OSErr;

| 12 | ioCompletion       | pointer |
| 16 | ioResult           | word    |
| 18 | ioNamePtr          | pointer |
| 22 | ioVRefNum          | word    |

FUNCTION PBFlushVol (paramBlock: ParmBlkPtr; async: BOOLEAN) : OSErr;

| 12 | ioCompletion       | pointer |
| 16 | ioResult           | word    |
| 18 | ioNamePtr          | pointer |
| 22 | ioVRefNum          | word    |

FUNCTION PBUnmountVol (paramBlock: ParmBlkPtr) : OSErr;

| 16 | ioResult           | word    |
| 18 | ioNamePtr          | pointer |
| 22 | ioVRefNum          | word    |

FUNCTION PBOffLine (paramBlock: ParmBlkPtr) : OSErr;

| 12 | ioCompletion       | pointer |
| 16 | ioResult           | word    |
| 18 | ioNamePtr          | pointer |
| 22 | ioVRefNum          | word    |
FUNCTION PBEject (paramBlock: ParmBlkPtr) : OSErr;
  → 12 ioCompletion pointer
  ← 16 ioResult word
  → 18 ioNamePtr pointer
  → 22 ioVRefNum word

Accessing Files

FUNCTION PBCreate (paramBlock: ParmBlkPtr; async: BOOLEAN) : OSErr;
  → 12 ioCompletion pointer
  ← 16 ioResult word
  → 18 ioNamePtr pointer
  → 22 ioVRefNum word
  → 26 ioFVersNum byte

FUNCTION PBOpen (paramBlock: ParmBlkPtr; async: BOOLEAN) : OSErr;
  → 12 ioCompletion pointer
  ← 16 ioResult word
  → 18 ioNamePtr pointer
  → 22 ioVRefNum word
  ← 24 ioRefNum word
  → 26 ioVersNum byte
  → 27 ioPermssn byte
  ← 28 ioMisc pointer

FUNCTION PBOpenRF (paramBlock: ParmBlkPtr; async: BOOLEAN) : OSErr;
  → 12 ioCompletion pointer
  ← 16 ioResult word
  → 18 ioNamePtr pointer
  → 22 ioVRefNum word
  ← 24 ioRefNum word
  → 26 ioVersNum byte
  → 27 ioPermssn byte
  → 28 ioMisc pointer

FUNCTION PBRead (paramBlock: ParmBlkPtr; async: BOOLEAN) : OSErr;
  → 12 ioCompletion pointer
  ← 16 ioResult word
  → 24 ioRefNum word
  ← 32 ioBuffer pointer
  → 36 ioReqCount long word
  ← 40 ioActCount long word
  → 44 ioPosMode word
  ← 46 ioPosOffset long word
FUNCTION PBWrite (paramBlock: ParmBlkPtr; async: BOOLEAN) : OSErr;
→ 12  ioCompletion  pointer
← 16  ioResult  word
→ 24  ioRefNum  word
→ 32  ioBuffer  pointer
→ 36  ioReqCount  long word
← 40  ioActCount  long word
→ 44  ioPosMode  word
← 46  ioPosOffset  long word

FUNCTION PBGetFPos (paramBlock: ParmBlkPtr; async: BOOLEAN) : OSErr;
→ 12  ioCompletion  pointer
← 16  ioResult  word
→ 24  ioRefNum  word
← 36  ioReqCount  long word
← 40  ioActCount  long word
← 44  ioPosMode  word
← 46  ioPosOffset  long word

FUNCTION PBSetFPos (paramBlock: ParmBlkPtr; async: BOOLEAN) : OSErr;
→ 12  ioCompletion  pointer
← 16  ioResult  word
→ 24  ioRefNum  word
→ 44  ioPosMode  word
← 46  ioPosOffset  long word

FUNCTION PBGetEOF (paramBlock: ParmBlkPtr; async: BOOLEAN) : OSErr;
→ 12  ioCompletion  pointer
← 16  ioResult  word
→ 24  ioRefNum  word
← 28  ioMisc  long word

FUNCTION PBSetEOF (paramBlock: ParmBlkPtr; async: BOOLEAN) : OSErr;
→ 12  ioCompletion  pointer
← 16  ioResult  word
→ 24  ioRefNum  word
← 28  ioMisc  long word

FUNCTION PBAllocate (paramBlock: ParmBlkPtr; async: BOOLEAN) : OSErr;
→ 12  ioCompletion  pointer
← 16  ioResult  word
→ 24  ioRefNum  word
→ 36  ioReqCount  long word
← 40  ioActCount  long word

FUNCTION PBFsushFile (paramBlock: ParmBlkPtr; async: BOOLEAN) : OSErr;
→ 12  ioCompletion  pointer
← 16  ioResult  word
→ 24  ioRefNum  word
FUNCTION PBClose (paramBlock: ParmBlkPtr; async: BOOLEAN) : OSErr;
   → 12  ioCompletion     pointer
   ← 16  ioResult        word
   → 24  ioRefNum       word

Changing Information About Files

FUNCTION PBGetFileInfo (paramBlock: ParmBlkPtr; async: BOOLEAN) : OSErr;
   → 12  ioCompletion    pointer
   ← 16  ioResult       word
   ← 18  ioNamePtr      pointer
   → 22  ioVRefNum   word
   → 24  ioFRefNum     word
   → 26  ioFVersNum    byte
   → 28  ioFDirIndex  word
   ← 30  ioFAttrib     byte
   ← 31  ioFVersNum    byte
   ← 32  ioFIndrInfo   16 bytes
   → 48  ioFNum        long word
   → 52  ioFStBlk      word
   → 54  ioFLen        long word
   → 58  ioFPyLen      long word
   ← 62  ioFIRSTBlk    word
   ← 64  ioFIRSTLen   long word
   ← 68  ioFIRSTPyLen long word
   ← 72  ioFIRAttrib  long word
   ← 76  ioFIRStBlk   long word

FUNCTION PBSetFileInfo (paramBlock: ParmBlkPtr; async: BOOLEAN) : OSErr;
   → 12  ioCompletion    pointer
   ← 16  ioResult       word
   ← 18  ioNamePtr      pointer
   → 22  ioVRefNum   word
   → 26  ioFVersNum    byte
   ← 32  ioFIndrInfo   16 bytes
   → 48  ioFNum        long word
   ← 52  ioFStBlk      word
   ← 54  ioFLen        long word
   ← 58  ioFPyLen      long word
   ← 62  ioFIRSTBlk    word
   ← 64  ioFIRSTLen   long word
   ← 68  ioFIRSTPyLen long word
   ← 72  ioFIRAttrib  long word
   ← 76  ioFIRStBlk   long word

FUNCTION PBSetFLock (paramBlock: ParmBlkPtr; async: BOOLEAN) : OSErr;
   → 12  ioCompletion    pointer
   ← 16  ioResult       word
   ← 18  ioNamePtr      pointer
   → 22  ioVRefNum   word
   → 26  ioFVersNum    byte

FUNCTION PBRestFLock (paramBlock: ParmBlkPtr; async: BOOLEAN) : OSErr;
   → 12  ioCompletion    pointer
   ← 16  ioResult       word
   ← 18  ioNamePtr      pointer
   → 22  ioVRefNum   word
   → 26  ioFVersNum    byte
FUNCTION PBSetFVers (paramBlock: ParmBlkPtr; async: BOOLEAN) : OSErr;
  → 12  ioCompletion  pointer
  ← 16  ioResult  word
  → 18  ioNamePtr  pointer
  → 22  ioVRefNum  word
  → 26  ioVersNum  byte
  → 28  ioMisc  byte

FUNCTION PBRename (paramBlock: ParmBlkPtr; async: BOOLEAN) : OSErr;
  → 12  ioCompletion  pointer
  ← 16  ioResult  word
  → 18  ioNamePtr  pointer
  → 22  ioVRefNum  word
  → 26  ioVersNum  byte
  → 28  ioMisc  pointer

FUNCTION PBDelete (paramBlock: ParmBlkPtr; async: BOOLEAN) : OSErr;
  → 12  ioCompletion  pointer
  ← 16  ioResult  word
  → 18  ioNamePtr  pointer
  → 22  ioVRefNum  word
  → 26  ioVersNum  byte

Accessing Queues  [Not in ROM]

FUNCTION GetFSQHdr : QHdrPtr;
FUNCTION GetVCBQHdr : QHdrPtr;
FUNCTION GetDrvQHdr : QHdrPtr;

Result Codes

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>badMDBErr</td>
<td>-60</td>
<td>Master directory block is bad; must reinitialize volume</td>
</tr>
<tr>
<td>bdNamErr</td>
<td>-37</td>
<td>Bad file name or volume name (perhaps zero-length)</td>
</tr>
<tr>
<td>dirFulErr</td>
<td>-33</td>
<td>File directory full</td>
</tr>
<tr>
<td>dskFulErr</td>
<td>-34</td>
<td>All allocation blocks on the volume are full</td>
</tr>
<tr>
<td>dupFNErr</td>
<td>-48</td>
<td>A file with the specified name and version number already exists</td>
</tr>
<tr>
<td>eofErr</td>
<td>-39</td>
<td>Logical end-of-file reached during read operation</td>
</tr>
<tr>
<td>extFSErr</td>
<td>-58</td>
<td>External file system; file-system identifier is nonzero, or path reference number is greater than 1024</td>
</tr>
<tr>
<td>fBsyErr</td>
<td>-47</td>
<td>One or more files are open</td>
</tr>
<tr>
<td>fLckdErr</td>
<td>-45</td>
<td>File locked</td>
</tr>
<tr>
<td>fnfErr</td>
<td>-43</td>
<td>File not found</td>
</tr>
<tr>
<td>fnOpnErr</td>
<td>-38</td>
<td>File not open</td>
</tr>
</tbody>
</table>

III-106  File Manager
Summary

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>fsRnErr</td>
<td>-59</td>
<td>Problem during rename</td>
</tr>
<tr>
<td>gfpErr</td>
<td>-52</td>
<td>Error during GetFPos</td>
</tr>
<tr>
<td>ioErr</td>
<td>-36</td>
<td>I/O error</td>
</tr>
<tr>
<td>memFullErr</td>
<td>-108</td>
<td>Not enough room in heap zone</td>
</tr>
<tr>
<td>noErr</td>
<td>0</td>
<td>No error</td>
</tr>
<tr>
<td>noMacDskErr</td>
<td>-57</td>
<td>Volume lacks Macintosh-format directory</td>
</tr>
<tr>
<td>nsDrvErr</td>
<td>-56</td>
<td>Specified drive number doesn't match any number in the drive queue</td>
</tr>
<tr>
<td>nsVErr</td>
<td>-35</td>
<td>Specified volume doesn't exist</td>
</tr>
<tr>
<td>opWrErr</td>
<td>-49</td>
<td>The read/write permission of only one access path to a file can allow writing</td>
</tr>
<tr>
<td>paramErr</td>
<td>-50</td>
<td>Parameters don't specify an existing volume, and there's no default volume</td>
</tr>
<tr>
<td>permErr</td>
<td>-54</td>
<td>Attempt to open locked file for writing</td>
</tr>
<tr>
<td>posErr</td>
<td>-40</td>
<td>Attempt to position before start of file</td>
</tr>
<tr>
<td>rfNumErr</td>
<td>-51</td>
<td>Reference number specifies nonexistent access path</td>
</tr>
<tr>
<td>tmfoErr</td>
<td>-42</td>
<td>Too many files open</td>
</tr>
<tr>
<td>volOffLinErr</td>
<td>-53</td>
<td>Volume not on-line</td>
</tr>
<tr>
<td>volOnLinErr</td>
<td>-55</td>
<td>Specified volume is already mounted and on-line</td>
</tr>
<tr>
<td>vLckdErr</td>
<td>-46</td>
<td>Volume is locked by a software flag</td>
</tr>
<tr>
<td>wrPermErr</td>
<td>-61</td>
<td>Read/write permission doesn't allow writing</td>
</tr>
<tr>
<td>wPrErr</td>
<td>-44</td>
<td>Volume is locked by a hardware setting</td>
</tr>
</tbody>
</table>

Assembly-Language Information

Constants

; Flags in file information used by the Finder

fHasBundle .EQU 13 ;set if file has a bundle
gInvisible .EQU 14 ;set if file's icon is invisible

; Flags in trap words

asyncTrpBit .EQU 10 ;set for an asynchronous call
noQueueBit .EQU 9 ;set for immediate execution
Values for requesting read/write access

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fsCurPerm</td>
<td>EQU</td>
<td>0; whatever is currently allowed</td>
</tr>
<tr>
<td>fsRdPerm</td>
<td>EQU</td>
<td>1; request to read only</td>
</tr>
<tr>
<td>fsWrPerm</td>
<td>EQU</td>
<td>2; request to write only</td>
</tr>
<tr>
<td>fsRdWrPerm</td>
<td>EQU</td>
<td>3; request to read and write</td>
</tr>
</tbody>
</table>

Positioning modes

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fsAtMark</td>
<td>EQU</td>
<td>0; at current mark</td>
</tr>
<tr>
<td>fsFromStart</td>
<td>EQU</td>
<td>1; offset relative to beginning of file</td>
</tr>
<tr>
<td>fsFromEOF</td>
<td>EQU</td>
<td>2; offset relative to logical end-of-file</td>
</tr>
<tr>
<td>fsFromMark</td>
<td>EQU</td>
<td>3; offset relative to current mark</td>
</tr>
<tr>
<td>rdVerify</td>
<td>EQU</td>
<td>64; add to above for read-verify</td>
</tr>
</tbody>
</table>

Structure of File Information Used by the Finder

- `fdType` File type (long)
- `fdCreator` File's creator (long)
- `fdFlags` Flags (word)
- `fdLocation` File's location (point; long)
- `fdFlr` File's window (word)

Standard Parameter Block Data Structure

- `qLink` Pointer to next queue entry
- `qType` Queue type (word)
- `ioTrap` Routine trap (word)
- `ioCmdAddr` Routine address
- `ioCompletion` Address of completion routine
- `ioResult` Result code (word)
- `ioFileName` Pointer to file name (preceded by length byte)
- `ioVNPtr` Pointer to volume name (preceded by length byte)
- `ioVRefNum` Volume reference number (word)
- `ioDrvNum` Drive number (word)

I/O Parameter Block Data Structure

- `ioRefNum` Path reference number (word)
- `ioFileType` Version number (byte)
- `ioPermsn` Read/write permission (byte)
- `ioNewName` Pointer to new file or volume name for Rename
- `ioLEOF` Logical end-of-file for SetEOF (long)
- `ioOwnBuf` Pointer to access path buffer
- `ioNewType` New version number for SetFileType (byte)
- `ioBuffer` Pointer to data buffer
- `ioReqCount` Requested number of bytes (long)
- `ioActCount` Actual number of bytes (long)
- `ioPosMode` Positioning mode and newline character (word)
ioPosOffset  Positioning offset (long)
ioQElSize    Size in bytes of I/O parameter block

Structure of File Information Parameter Block

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ioRefNum</td>
<td>Path reference number (word)</td>
</tr>
<tr>
<td>ioFileType</td>
<td>Version number (byte)</td>
</tr>
<tr>
<td>ioFDlndex</td>
<td>Sequence number of file (word)</td>
</tr>
<tr>
<td>ioFlAttrib</td>
<td>File attributes (byte)</td>
</tr>
<tr>
<td>ioFFIType</td>
<td>Version number (byte)</td>
</tr>
<tr>
<td>ioFLUsrWds</td>
<td>Information used by the Finder (16 bytes)</td>
</tr>
<tr>
<td>ioFFINum</td>
<td>File number (long)</td>
</tr>
<tr>
<td>ioFIlStBlk</td>
<td>First allocation block of data fork (word)</td>
</tr>
<tr>
<td>ioFLgLen</td>
<td>Logical end-of-file of data fork (long)</td>
</tr>
<tr>
<td>ioFLPyLen</td>
<td>Physical end-of-file of data fork (long)</td>
</tr>
<tr>
<td>ioFIRStBlk</td>
<td>First allocation block of resource fork (word)</td>
</tr>
<tr>
<td>ioFIRgLen</td>
<td>Logical end-of-file of resource fork (long)</td>
</tr>
<tr>
<td>ioFIRPyLen</td>
<td>Physical end-of-file of resource fork (long)</td>
</tr>
<tr>
<td>ioFlCrDat</td>
<td>Date and time of creation (long)</td>
</tr>
<tr>
<td>ioFlMdDat</td>
<td>Date and time of last modification (long)</td>
</tr>
<tr>
<td>ioFQEISize</td>
<td>Size in bytes of file information parameter block</td>
</tr>
</tbody>
</table>

Structure of Volume Information Parameter Block

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ioVollndex</td>
<td>Volume index (word)</td>
</tr>
<tr>
<td>ioVCrDate</td>
<td>Date and time of initialization (long)</td>
</tr>
<tr>
<td>ioVLsBkUp</td>
<td>Date and time of last backup (long)</td>
</tr>
<tr>
<td>ioVAtrb</td>
<td>Volume attributes (word)</td>
</tr>
<tr>
<td>ioVNmFls</td>
<td>Number of files in directory (word)</td>
</tr>
<tr>
<td>ioVDirSt</td>
<td>First block of directory (word)</td>
</tr>
<tr>
<td>ioVBlLn</td>
<td>Length of directory in blocks (word)</td>
</tr>
<tr>
<td>ioVNmAlBlks</td>
<td>Number of allocation blocks on volume (word)</td>
</tr>
<tr>
<td>ioVAIBlkSiz</td>
<td>Size of allocation blocks (long)</td>
</tr>
<tr>
<td>ioVCipSiz</td>
<td>Number of bytes to allocate (long)</td>
</tr>
<tr>
<td>ioAIBlSt</td>
<td>First allocation block in block map (word)</td>
</tr>
<tr>
<td>ioVNxtFNum</td>
<td>Next unused file number (long)</td>
</tr>
<tr>
<td>ioVFrBlk</td>
<td>Number of unused allocation blocks (word)</td>
</tr>
<tr>
<td>ioVQEISize</td>
<td>Size in bytes of volume information parameter block</td>
</tr>
</tbody>
</table>

Volume Information Data Structure

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>drSi gWord</td>
<td>Always $D2D7 (word)</td>
</tr>
<tr>
<td>drCrDate</td>
<td>Date and time of initialization (long)</td>
</tr>
<tr>
<td>drLsBkUp</td>
<td>Date and time of last backup (long)</td>
</tr>
<tr>
<td>drAtrb</td>
<td>Volume attributes (word)</td>
</tr>
<tr>
<td>drNmFls</td>
<td>Number of files in directory (word)</td>
</tr>
<tr>
<td>drDirSt</td>
<td>First block of directory (word)</td>
</tr>
<tr>
<td>drBLn</td>
<td>Length of directory in blocks (word)</td>
</tr>
<tr>
<td>drNmAlBlks</td>
<td>Number of allocation blocks on volume (word)</td>
</tr>
<tr>
<td>drAlBlkSiz</td>
<td>Size of allocation blocks (long)</td>
</tr>
</tbody>
</table>
Inside Macintosh

drClpSiz  Number of bytes to allocate (long)
drALBlSt  First allocation block in block map (word)
drNxtFNNum Next unused file number (long)
drFreeBks Number of unused allocation blocks (word)
drVN     Volume name preceded by length byte (28 bytes)

File Directory Entry Data Structure

flags    Bit 7=1 if entry used; bit 0=1 if file locked (byte)
flTyp     Version number (byte)
flUsrWds  Information used by the Finder (16 bytes)
flFNNum   File number (long)
flStBlk   First allocation block of data fork (word)
flLgLen   Logical end-of-file of data fork (long)
flPyLen   Physical end-of-file of data fork (long)
flRStBlk  First allocation block of resource fork (word)
flRLgLen  Logical end-of-file of resource fork (long)
flRPyLen  Physical end-of-file of resource fork (long)
flCrDat   Date and time file of creation (long)
flMdDat   Date and time of last modification (long)
flNam     File name preceded by length byte

Volume Control Block Data Structure

qLink     Pointer to next queue entry
qType     Queue type (word)
vcbFlags  Bit 15=1 if volume control block is dirty (word)
vcbSigWord Always $D2D7 (word)
vcbCrDate Date and time of initialization (word)
vcbLsBkUp Date and time of last backup (long)
vcbAtrb   Volume attributes (word)
vcbNmFls  Number of files in directory (word)
vcbDirSt  First block of directory (word)
vcbBILn   Length of directory in blocks (word)
vcbNmBlks Number of allocation blocks on volume (word)
vcbAlBlkSiz Size of allocation blocks (long)
vcbClpSiz Number of bytes to allocate (long)
vcbAlBlSt First allocation block in block map (word)
vcbNxtFNNum Next unused file number (long)
vcbFreeBks Number of unused allocation blocks (word)
vcbVN     Volume name preceded by length byte (28 bytes)
vcbDrvNum Drive number of drive in which volume is mounted (word)
vcbDRenNum Driver reference number of driver for drive in which volume is mounted (word)
vcbFSID   File-system identifier (word)
vcbVRefNum Volume reference number (word)
vcbMAadr  Pointer to volume block map
vcbBufAdr  Pointer to volume buffer
vcbMLen   Number of bytes in volume block map (word)

III-110 File Manager
### File Control Block Data Structure

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fcbFINum</td>
<td>File number (long)</td>
</tr>
<tr>
<td>fcbMdRByt</td>
<td>Flags (byte)</td>
</tr>
<tr>
<td>fcbTypByt</td>
<td>Version number (byte)</td>
</tr>
<tr>
<td>fcbSBlk</td>
<td>First allocation block of file (word)</td>
</tr>
<tr>
<td>fcbEOF</td>
<td>Logical end-of-file (long)</td>
</tr>
<tr>
<td>fcbPLen</td>
<td>Physical end-of-file (long)</td>
</tr>
<tr>
<td>fcbCrPs</td>
<td>Mark (long)</td>
</tr>
<tr>
<td>fcbVPtr</td>
<td>Pointer to volume control block (long)</td>
</tr>
<tr>
<td>fcbBfAdr</td>
<td>Pointer to access path buffer (long)</td>
</tr>
</tbody>
</table>

### Drive Queue Entry Data Structure

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>qLink</td>
<td>Pointer to next queue entry</td>
</tr>
<tr>
<td>qType</td>
<td>Queue type (word)</td>
</tr>
<tr>
<td>dQDrive</td>
<td>Drive number (word)</td>
</tr>
<tr>
<td>dQRefNum</td>
<td>Driver reference number (word)</td>
</tr>
<tr>
<td>dQFSID</td>
<td>File-system identifier (word)</td>
</tr>
<tr>
<td>dQDrvSize</td>
<td>Number of logical blocks (word)</td>
</tr>
</tbody>
</table>

### Macro Names

<table>
<thead>
<tr>
<th>Pascal name</th>
<th>Macro name</th>
</tr>
</thead>
<tbody>
<tr>
<td>FInitQueue</td>
<td>_InitQueue</td>
</tr>
<tr>
<td>PBMountVol</td>
<td>_MountVol</td>
</tr>
<tr>
<td>PBGetVInfo</td>
<td>_GetVInfo</td>
</tr>
<tr>
<td>PBGetVol</td>
<td>_GetVol</td>
</tr>
<tr>
<td>PBSetVol</td>
<td>_SetVol</td>
</tr>
<tr>
<td>PBFlushVol</td>
<td>_FlushVol</td>
</tr>
<tr>
<td>PBUnmountVol</td>
<td>_UnmountVol</td>
</tr>
<tr>
<td>PBOffLine</td>
<td>_OffLine</td>
</tr>
<tr>
<td>PEject</td>
<td>_Eject</td>
</tr>
<tr>
<td>PBCreate</td>
<td>_Create</td>
</tr>
<tr>
<td>PBOpen</td>
<td>_Open</td>
</tr>
<tr>
<td>PBOpenRF</td>
<td>_OpenRF</td>
</tr>
<tr>
<td>PBRead</td>
<td>_Read</td>
</tr>
<tr>
<td>PWrite</td>
<td>_Write</td>
</tr>
<tr>
<td>PBGetFPos</td>
<td>_GetFPos</td>
</tr>
<tr>
<td>PBSetFPos</td>
<td>_SetFPos</td>
</tr>
<tr>
<td>PBGetEOF</td>
<td>_GetEOF</td>
</tr>
<tr>
<td>PBSetEOF</td>
<td>_SetEOF</td>
</tr>
<tr>
<td>PBAllocate</td>
<td>_Allocate</td>
</tr>
<tr>
<td>PBFlushFile</td>
<td>_FlushFile</td>
</tr>
<tr>
<td>PClose</td>
<td>_Close</td>
</tr>
<tr>
<td>PBGetFInfo</td>
<td>_GetFInfo</td>
</tr>
<tr>
<td>PBSetFInfo</td>
<td>_SetFInfo</td>
</tr>
<tr>
<td>PBSetFLock</td>
<td>_SetFLock</td>
</tr>
<tr>
<td>PBRstFLock</td>
<td>_RstFLock</td>
</tr>
</tbody>
</table>
**Inside Macintosh**

<table>
<thead>
<tr>
<th>PBSetFVrs</th>
<th>_SetFilType</th>
</tr>
</thead>
<tbody>
<tr>
<td>PBRename</td>
<td>_Rename</td>
</tr>
<tr>
<td>PBDelete</td>
<td>_Delete</td>
</tr>
</tbody>
</table>

**Variables**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FSQHdr</td>
<td>File I/O queue header (10 bytes)</td>
</tr>
<tr>
<td>VCBQHdr</td>
<td>Volume-control-block queue header (10 bytes)</td>
</tr>
<tr>
<td>DefVCBPtr</td>
<td>Pointer to default volume control block</td>
</tr>
<tr>
<td>FCBSPtr</td>
<td>Pointer to file-control-block buffer</td>
</tr>
<tr>
<td>DrvQHdr</td>
<td>Drive queue header (10 bytes)</td>
</tr>
<tr>
<td>ToExtFS</td>
<td>Pointer to external file system</td>
</tr>
</tbody>
</table>

*III-112 File Manager*
FONT MANAGER

Constants

CONST { Font numbers }

    systemFont = 0;  {system font}
    applFont = 1;   {application font}
    newYork = 2;
    geneva = 3;
    monaco = 4;
    venice = 5;
    london = 6;
    athens = 7;
    sanFran = 8;
    toronto = 9;
    cairo = 11;
    losAngeles = 12;
    times = 20;
    helvetica = 21;
    courier = 22;
    symbol = 23;
    taliesin = 24;

    { Special characters }

    commandMark = $11;  {Command key symbol}
    checkMark = $12;   {check mark}
    diamondMark = $13; {diamond symbol}
    appleMark = $14;   {apple symbol}

    { Font types }

    propFont = $9000;  {proportional font}
    fixedFont = $8000; {fixed-width font}
    fontWid = $ACB0;   {font width data}

Data Types

TYPE FMInput = PACKED RECORD

    family: INTEGER;  {font number}
    size: INTEGER;   {font size}
    face: Style;     {character style}
    needBits: BOOLEAN;  {TRUE if drawing}
    device: INTEGER; {device-specific information}
    numer: Point;    {numerators of scaling factors}
    denom: Point     {denominators of scaling factors}

END;
Inside Macintosh

FMOutPtr = ^FMOutput;
FMOutput =

PACKED RECORD
errNum: INTEGER; {not used}
fontHandle: Handle; {handle to font record}
bold: Byte; {bold factor}
italic: Byte; {italic factor}
ulOffset: Byte; {underline offset}
ulShadow: Byte; {underline shadow}
ulThick: Byte; {underline thickness}
shadow: Byte; {shadow factor}
extra: SignedByte; {width of style}
ascent: Byte; {ascent}
descent: Byte; {descent}
widMax: Byte; {maximum character width}
leading: SignedByte; {leading}
unused: Byte; {not used}
umer: Point; {numerators of scaling factors}
denom: Point; {denominators of scaling factors}
END;

FontRec =
RECORD
fontType: INTEGER; {font type}
firstChar: INTEGER; {ASCII code of first character}
lastChar: INTEGER; {ASCII code of last character}
widMax: INTEGER; {maximum character width}
kernMax: INTEGER; {negative of maximum character kern}
nDescent: INTEGER; {negative of descent}
frRectWidth: INTEGER; {width of font rectangle}
frRectHeight: INTEGER; {height of font rectangle}
owTLoc: INTEGER; {offset to offset/width table}
ascent: INTEGER; {ascent}
descent: INTEGER; {descent}
leading: INTEGER; {leading}
rowWords: INTEGER; {row width of bit image / 2}
{bitImage: ARRAY[1..rowWords,1..frRectHeight] OF INTEGER; }
{bit image}
{locTable: ARRAY[firstChar..lastChar+2] OF INTEGER; }
{location table}
{owTable: ARRAY[firstChar..lastChar+2] OF INTEGER; }
{offset/width table}
END;

Routines

Initializing the Font Manager

PROCEDURE InitFonts;

III-114 Font Manager
Getting Font Information

PROCEDURE GetFontName (fontNum: INTEGER; VAR theName: Str255);
PROCEDURE GetFNum (fontName: Str255; VAR theNum: INTEGER);
FUNCTION RealFont (fontNum: INTEGER; size: INTEGER): BOOLEAN;

Keeping Fonts in Memory

PROCEDURE SetFontLock (lockFlag: BOOLEAN);

Advanced Routine

FUNCTION FMSwapFont (inRec: FMInput): FMOutPtr;

Assembly-Language Information

Constants

; Font numbers
sysFont .EQU 0 ; system font
applFont .EQU 1 ; application font
newYork .EQU 2
geneva .EQU 3
monaco .EQU 4
venice .EQU 5
london .EQU 6
athens .EQU 7
sanFran .EQU 8
toronto .EQU 9
cairo .EQU 11
losAngeles .EQU 12	
times .EQU 20
helvetica .EQU 21
courier .EQU 22
symbol .EQU 23
taliesin .EQU 24

; Special characters
commandMark .EQU $11 ; Command key symbol
checkMark .EQU $12 ; check mark
diamondMark .EQU $13 ; diamond symbol
appleMark .EQU $14 ; apple symbol

; Font types
propFont .EQU $9000 ; proportional font
Inside Macintosh

fixedFont .EQU $B000 ;fixed-width font
fontWid .EQU $ACB0 ;font width data

; Control and status call code
fMgrCtl1 .EQU 8 ;code used to get and modify font
; characterization table

Font Input Record Data Structure

fmInFamily Font number (word)
fInSize Font size (word)
fInFace Character style (word)
fInNeedBits Nonzero if drawing (byte)
fInDevice Device-specific information (byte)
fInNumer Numerators of scaling factors (point; long)
fInDenom Denominators of scaling factors (point; long)

Font Output Record Data Structure

fmOutFontH Handle to font record
fmOutBold Bold factor (byte)
fOutItalic Italic factor (byte)
fOutULOffset Underline offset (byte)
fOutULShadow Underline shadow (byte)
fOutULThick Underline thickness (byte)
fOutShadow Shadow factor (byte)
fOutExtra Width of style (byte)
fOutAscent Ascent (byte)
fOutDescent Descent (byte)
fOutWidMax Maximum character width (byte)
fOutLeading Leading (byte)
fOutNumer Numerators of scaling factors (point; long)
fOutDenom Denominators of scaling factors (point; long)

Font Record Data Structure

fFontType Font type (word)
fFirstChar ASCII code of first character (word)
fLastChar ASCII code of last character (word)
fWidMax Maximum character width (word)
fKernMax Negative of maximum character kern (word)
fnDescent Negative of descent (word)
fRectWidth Width of font rectangle (word)
fRectHeight Height of font rectangle (word)
fOWTLoc Offset to offset/width table (word)
Ascent Ascent (word)
fDescent Descent (word)
Leading Leading (word)
fRowWords  Row width of bit image / 2 (word)

**Special Macro Names**

<table>
<thead>
<tr>
<th>Pascal name</th>
<th>Macro name</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetFontName</td>
<td>_GetFName</td>
</tr>
</tbody>
</table>

**Variables**

- **ApFontID**
  - Font number of application font (word)
- **FScaleDisable**
  - Nonzero to disable scaling (byte)
- **ROMFont0**
  - Handle to font record for system font
INTERNATIONAL UTILITIES PACKAGE

Constants

CONST { Masks for currency format }

currSymLead = 16;  {set if currency symbol leads}
currNegSym = 32;  {set if minus sign for negative}
currTrailingZ = 64;  {set if trailing decimal zeroes}
currLeadingZ = 128;  {set if leading integer zero}

{ Order of short date elements }

mdy = 0;  {month day year}
dmy = 1;  {day month year}
ymd = 2;  {year month day}

{ Masks for short date format }

dayLeadingZ = 32;  {set if leading zero for day}
mntLeadingZ = 64;  {set if leading zero for month}
century = 128;  {set if century included}

{ Masks for time format }

secLeadingZ = 32;  {set if leading zero for seconds}
minLeadingZ = 64;  {set if leading zero for minutes}
hrLeadingZ = 128;  {set if leading zero for hours}

{ High-order byte of version information }

verUS = 0;
verFrance = 1;
verBritain = 2;
verGermany = 3;
verItaly = 4;
verNetherlands = 5;
verBelgiumLux = 6;
verSweden = 7;
verSpain = 8;
verDenmark = 9;
verPortugal = 10;
verFrCanada = 11;
verNorway = 12;
verIsrael = 13;
verJapan = 14;
verAustralia = 15;
verArabia = 16;
verFinland = 17;

III-118 International Utilities Package
Data Types

TYPE Int10Hndl = ^Int10Ptr;
Int10Ptr = ^Int10Rec;
Int10Rec =

PACKED RECORD
  decimalPt: CHAR; {decimal point character}
  thousSep: CHAR; {thousands separator}
  listSep: CHAR; {list separator}
  currSym1: CHAR; {currency symbol}
  currSym2: CHAR;
  currSym3: CHAR;
  currFmt: Byte; {currency format}
  dateOrder: Byte; {order of short date elements}
  shrtDateFmt: Byte; {short date format}
  dateSep: CHAR; {date separator}
  timeCycle: Byte; {0 if 24-hour cycle, 255 if 12-hour}
  timeFmt: Byte; {time format}
  mornStr: PACKED ARRAY[1..4] OF CHAR;
    {trailing string for first 12-hour cycle}
  eveStr: PACKED ARRAY[1..4] OF CHAR;
    {trailing string for last 12-hour cycle}
  timeSep: CHAR; {time separator}
  time1Suff: CHAR; {trailing string for 24-hour cycle}
  time2Suff: CHAR;
  time3Suff: CHAR;
  time4Suff: CHAR;
  time5Suff: CHAR;
  time6Suff: CHAR;
  time7Suff: CHAR;
  time8Suff: CHAR;
  metricSys: Byte; {255 if metric, 0 if not}
  int10Vers: INTEGER {version information}
END;
Inside Macintosh

Int11Hndl = ^Int11Ptr;
Int11Ptr = ^Int11Rec;
Int11Rec =

PACKED RECORD
days: ARRAY[1..7] OF STRING[15];  {day names}
months: ARRAY[1..12] OF STRING[15];  {month names}
suppressDay: Byte;  {0 for day name, 255 for none}
lngDateFmt: Byte;  {order of long date elements}
dayLeading0: Byte;  {255 for leading 0 in day number}
abbrLen: Byte;  {length for abbreviating names}
st0: PACKED ARRAY[1..4] OF CHAR;  {strings }
st1: PACKED ARRAY[1..4] OF CHAR;  { for }
st2: PACKED ARRAY[1..4] OF CHAR;  { long }
st3: PACKED ARRAY[1..4] OF CHAR;  { date }
st4: PACKED ARRAY[1..4] OF CHAR;  { format}
int11Vers: INTEGER;  {version information}
localRtn: INTEGER  {routine for localizing string }  
            { comparison; actually may be }  
            { longer than one integer}
END;

DateForm = (shortDate, longDate, abbrevDate);

Routines

PROCEDURE IUDateString  (dateTime: LONGINT; form: DateForm; VAR result: Str255);
PROCEDURE IUDatePString  (dateTime: LONGINT; form: DateForm; VAR result: Str255; intlParam: Handle);
PROCEDURE IUTimeString   (dateTime: LONGINT; wantSeconds: BOOLEAN; VAR result: Str255);
PROCEDURE IUTimePString  (dateTime: LONGINT; wantSeconds: BOOLEAN; VAR result: Str255; intlParam: Handle);
FUNCTION IUMetric       : BOOLEAN;
FUNCTION IUGetIntl      (theID: INTEGER) : Handle;
PROCEDURE IUSetIntl     (refNum: INTEGER; theID: INTEGER; intlParam: Handle);
FUNCTION IUCmpString    (aStr,bStr: Str255) : INTEGER;  [NotinROM]
FUNCTION IUMagString    (aPtr,bPtr: Ptr; aLen,bLen: INTEGER) : INTEGER;
FUNCTION IUEqualString  (aStr,bStr: Str255) : INTEGER;  [NotinROM]
FUNCTION IUMagIDString  (aPtr,bPtr: Ptr; aLen,bLen: INTEGER) : INTEGER;

III-120 International Utilities Package
Assembly-Language Information

Constants

; Masks for currency format

currSymLead .EQU 16 ; set if currency symbol leads
currNegSym .EQU 32 ; set if minus sign for negative
currTrailingZ .EQU 64 ; set if trailing decimal zeroes
currLeadingZ .EQU 128 ; set if leading integer zero

; Order of short date elements

mdy .EQU 0 ; month day year
dmy .EQU 1 ; day month year
ymd .EQU 2 ; year month day

; Masks for short date format

dayLeadingZ .EQU 32 ; set if leading zero for day
mntLeadingZ .EQU 64 ; set if leading zero for month
century .EQU 128 ; set if century included

; Masks for time format

secLeadingZ .EQU 32 ; set if leading zero for seconds
minLeadingZ .EQU 64 ; set if leading zero for minutes
hrLeadingZ .EQU 128 ; set if leading zero for hours

; High-order byte of version information

verUS .EQU 0
verFrance .EQU 1
verBritain .EQU 2
verGermany .EQU 3
verItaly .EQU 4
verNetherlands .EQU 5
verBelgiumLux .EQU 6
verSweden .EQU 7
verSpain .EQU 8
verDenmark .EQU 9
verPortugal .EQU 10
verFrCanada .EQU 11
verNorway .EQU 12
verIsrael .EQU 13
verJapan .EQU 14
verAustralia .EQU 15
verArabia .EQU 16
verFinland .EQU 17
verFrSwiss .EQU 18
verGrSwiss .EQU 19
Inside Macintosh

verGreece .EQU 20
verIceland .EQU 21
verMalta .EQU 22
verCyprus .EQU 23
verTurkey .EQU 24
verYugoslavia .EQU 25

; Date form for IUDateString and IUDatePString
shortDate .EQU 0 ; short form of date
longDate .EQU 1 ; long form of date
abbrevDate .EQU 2 ; abbreviated long form

; Routine selectors
iuDateString .EQU 0
iuTimeString .EQU 2
iuMetric .EQU 4
iuGetIntl .EQU 6
iuSetIntl .EQU 8
iuMagString .EQU 10
iuMagIDString .EQU 12
iuDatePString .EQU 14
iuTimePString .EQU 16

International Resource 0 Data Structure

decimalPt Decimal point character (byte)
thousSep Thousands separator (byte)
listSep List separator (byte)
currSym Currency symbol (3 bytes)
currFmt Currency format (byte)
dateOrder Order of short date elements (byte)
shrtDateFmt Short date format (byte)
dateSep Date separator (byte)
timeCycle 0 if 24-hour cycle, 255 if 12-hour (byte)
timeFmt Time format (byte)
mornStr Trailing string for first 12-hour cycle (long)
eveStr Trailing string for last 12-hour cycle (long)
timeSep Time separator (byte)
timeSuff Trailing string for 24-hour cycle (8 bytes)
mintSys 255 if metric, 0 if not (byte)
intl0Vers Version information (word)

International Resource 1 Data Structure

days Day names (112 bytes)
months Month names (192 bytes)
suppressDay 0 for day name, 255 for none (byte)
lngDateFmt Order of long date elements (byte)
<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dayLeading0</td>
<td>255 for leading 0 in day number (byte)</td>
</tr>
<tr>
<td>abbrLen</td>
<td>Length for abbreviating names (byte)</td>
</tr>
<tr>
<td>st0</td>
<td>Strings for long date format (longs)</td>
</tr>
<tr>
<td>st1</td>
<td></td>
</tr>
<tr>
<td>st2</td>
<td></td>
</tr>
<tr>
<td>st3</td>
<td></td>
</tr>
<tr>
<td>st4</td>
<td></td>
</tr>
<tr>
<td>intl1Vers</td>
<td>Version information (word)</td>
</tr>
<tr>
<td>localRtn</td>
<td>Comparison localization routine</td>
</tr>
</tbody>
</table>

**Trap Macro Name**

_Pack6_
MEMORY MANAGER

Constants

CONST (Result codes)

memFullErr = -108;  {not enough room in heap zone}
memLockedErr = -117;  {block is locked}
memPurErr = -112;  {attempt to purge a locked block}
memWZErr = -111;  {attempt to operate on a free block}
nilHandleErr = -109;  {NIL master pointer}
noErr = 0;  {no error}

Data Types

TYPE SignedByte = -128..127;
Byte = 0..255;
Ptr = ^SignedByte;
Handle = ^Ptr;
Str255 = STRING[255];
StringPtr = ^Str255;
StringHandle = ^StringPtr;
ProcPtr = Ptr;
Fixed = LONGINT;
Size = LONGINT;
THz = ^Zone;
Zone = RECORD
  bkLim:      Ptr;     {zone trailer block}
  purgePtr:   Ptr;     {used internally}
  hFstFree:   Ptr;     {first free master pointer}
  zcbFree:    LONGINT; {number of free bytes}
  gzProc:     ProcPtr;    {grow zone function}
  moreMast:   INTEGER;  {master pointers to allocate}
  flags:      INTEGER;  {used internally}
  cntRel:     INTEGER;  {not used}
  maxRel:     INTEGER;  {not used}
  cntNRel:    INTEGER;  {not used}
  maxNRel:    INTEGER;  {not used}
  cntEmpty:   INTEGER;  {not used}
  cntHandles: INTEGER;  {not used}
  minCBFree:  LONGINT; {not used}
  purgeProc:  ProcPtr; {purge warning procedure}
  sparePtr:   Ptr;       {used internally}
  allocPtr:   Ptr;       {used internally}
  heapData:   INTEGER    {first usable byte in zone}
END;

III-124 Memory Manager
Routines

Initialization and Allocation

PROCEDURE InitApplZone;
PROCEDURE SetApplBase (startPtr: Ptr);
PROCEDURE InitZone (pGrowZone: ProcPtr; cMoreMasters: INTEGER;
limitPtr, startPtr: Ptr);
FUNCTION GetApplLimit : Ptr; [Not in ROM]
PROCEDURE SetApplLimit (zoneLimit:Ptr);
PROCEDURE MaxApplZone; [Not in ROM]
PROCEDURE MoreMasters;

Heap Zone Access

FUNCTION GetZone : THz;
PROCEDURE SetZone (hz: THz);
FUNCTION SystemZone : THz; [Not in ROM]
FUNCTION ApplicZone : THz; [Not in ROM]

Allocating and Releasing Relocatable Blocks

FUNCTION NewHandle (logicalSize: Size) : Handle;
PROCEDURE DisposHandle (h: Handle);
FUNCTION GetHandleSize (h: Handle) : Size;
PROCEDURE SetHandleSize (h: Handle; newSize: Size);
FUNCTION HandleZone (h: Handle) : THz;
FUNCTION RecoverHandle (p: Ptr) : Handle;
PROCEDURE ReallocHandle (h: Handle; logicalSize: Size);

Allocating and Releasing Nonrelocatable Blocks

FUNCTION NewPtr (logicalSize: Size) : Ptr;
PROCEDURE DisposPtr (p: Ptr);
FUNCTION GetPtrSize (p: Ptr) : Size;
PROCEDURE SetPtrSize (p: Ptr; newSize: Size);
FUNCTION PtrZone (p: Ptr) : THz;

Freeing Space in the Heap

FUNCTION FreeMem : LONGINT;
FUNCTION MaxMem (VAR grow: Size) : Size;
FUNCTION CompactMem (cbNeeded: Size) : Size;
PROCEDURE ResrvMem (cbNeeded: Size);
PROCEDURE PurgeMem (cbNeeded: Size);
PROCEDURE EmptyHandle (h: Handle);
Properties of Relocatable Blocks

PROCEDURE HLock (h: Handle);
PROCEDURE HUnlock (h: Handle);
PROCEDURE HPurge (h: Handle);
PROCEDURE HNoPurge (h: Handle);

Grow Zone Operations

PROCEDURE SetGrowZone (growZone: ProcPtr);
FUNCTION GZSaveHnd : Handle; [Not in ROM]

Miscellaneous Routines

PROCEDURE BlockMove (sourcePtr, destPtr: Ptr; byteCount: Size);
FUNCTION TopMem : Ptr; [Not in ROM]
PROCEDURE MoveHHi (h: Handle); [Not in ROM]
FUNCTION MemError : OSErr; [Not in ROM]

Grow Zone Function

FUNCTION MyGrowZone (cbNeeded: Size) : LONGINT;

Assembly-Language Information

Constants

; Values for tag byte of a block header

| tyBkFree  | EQU 0 | free block |
| tyBkNRel | EQU 1 | nonrelocatable block |
| tyBkRel   | EQU 2 | relocatable block |

; Flags for the high-order byte of a master pointer

| lock | EQU 7 | lock bit |
| purge | EQU 6 | purge bit |
| resourc | EQU 5 | resource bit |

; Result codes

| memFullErr | EQU -108 | not enough room in heap zone |
| memLockedErr | EQU -117 | block is locked |
| memPurErr | EQU -112 | attempt to purge a locked block |
| memWErr | EQU -111 | attempt to operate on a free block |
| nilHandleErr | EQU -109 | NIL master pointer |
| noErr | EQU 0 | no error |
### Zone Record Data Structure

- **bkLim**: Pointer to zone trailer block
- **hFstFree**: Pointer to first free master pointer
- **zcbFree**: Number of free bytes (long)
- **gzProc**: Address of grow zone function
- **mAllocCnt**: Master pointers to allocate (word)
- **purgeProc**: Address of purge warning procedure
- **heapData**: First usable byte in zone

### Block Header Data Structure

- **tagBC**: Tag byte and physical block size (long)
- **handle**: Relocatable block: relative handle
  - Nonrelocatable block: zone pointer
- **blkData**: First byte of block contents

### Parameter Block Structure for InitZone

- **startPtr**: Pointer to first byte in zone
- **limitPtr**: Pointer to first byte beyond end of zone
- **cMoreMasters**: Number of master pointers for zone (word)
- **pGrowZone**: Address of grow zone function

### Routines

<table>
<thead>
<tr>
<th>Trap macro</th>
<th>On entry</th>
<th>On exit</th>
</tr>
</thead>
<tbody>
<tr>
<td>_InitApplZone</td>
<td>A0: startPtr (ptr)</td>
<td>D0: result code (word)</td>
</tr>
<tr>
<td>_SetApplBase</td>
<td></td>
<td></td>
</tr>
<tr>
<td>_InitZone</td>
<td>A0: ptr to parameter block</td>
<td>D0: result code (word)</td>
</tr>
<tr>
<td></td>
<td>0 startPtr (ptr)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4 limitPtr (ptr)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>8 cMoreMasters (word)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>10 pGrowZone (ptr)</td>
<td></td>
</tr>
<tr>
<td>_SetApplLimit</td>
<td>A0: zoneLimit (ptr)</td>
<td>D0: result code (word)</td>
</tr>
<tr>
<td>_MoreMasters</td>
<td></td>
<td></td>
</tr>
<tr>
<td>_GetZone</td>
<td>A0: hz (ptr)</td>
<td>D0: result code (word)</td>
</tr>
<tr>
<td>_SetZone</td>
<td>D0: logicalSize (long)</td>
<td>A0: function result (handle)</td>
</tr>
<tr>
<td>_NewHandle</td>
<td></td>
<td></td>
</tr>
<tr>
<td>_DisposHandle</td>
<td>A0: h (handle)</td>
<td>D0: result code (word)</td>
</tr>
<tr>
<td>Trap macro</td>
<td>On entry</td>
<td>On exit</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------------------------------</td>
<td>----------------------------------------------</td>
</tr>
<tr>
<td>_GetHandleSize</td>
<td>A0: h (handle)</td>
<td>D0: if &gt;=0, function result (long)</td>
</tr>
<tr>
<td></td>
<td>D0: &gt;=0, result code (word)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>if &lt;0, result code (word)</td>
</tr>
<tr>
<td>_SetHandleSize</td>
<td>A0: h (handle)</td>
<td>D0: result code (word)</td>
</tr>
<tr>
<td></td>
<td>D0: newSize (long)</td>
<td></td>
</tr>
<tr>
<td>_HandleZone</td>
<td>A0: h (handle)</td>
<td>A0: function result (ptr)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>D0: result code (word)</td>
</tr>
<tr>
<td>_RecoverHandle</td>
<td>A0: p (ptr)</td>
<td>A0: function result (handle)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>D0: unchanged</td>
</tr>
<tr>
<td>_ReallocHandle</td>
<td>A0: h (handle)</td>
<td>D0: result code (word)</td>
</tr>
<tr>
<td></td>
<td>D0: logicalSize (long)</td>
<td></td>
</tr>
<tr>
<td>_NewPtr</td>
<td>D0: logicalSize (long)</td>
<td>A0: function result (ptr)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>D0: result code (word)</td>
</tr>
<tr>
<td>_DisposPtr</td>
<td>A0: p (ptr)</td>
<td>D0: result code (word)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>_GetPtrSize</td>
<td>A0: p (ptr)</td>
<td>D0: if &gt;=0, function result (long)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>if &lt;0, result code (word)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>A0: p (ptr)</td>
<td>D0: result code (word)</td>
</tr>
<tr>
<td>_SetPtrSize</td>
<td>A0: p (ptr)</td>
<td>D0: result code (word)</td>
</tr>
<tr>
<td></td>
<td>D0: newSize (long)</td>
<td></td>
</tr>
<tr>
<td>_PtrZone</td>
<td>A0: p (ptr)</td>
<td>A0: function result (ptr)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>D0: result code (word)</td>
</tr>
<tr>
<td>_FreeMem</td>
<td></td>
<td>D0: function result (long)</td>
</tr>
<tr>
<td>_MaxMem</td>
<td></td>
<td>D0: function result (long)</td>
</tr>
<tr>
<td></td>
<td>A0: grow (long)</td>
<td></td>
</tr>
<tr>
<td>_CompactMem</td>
<td>D0: cbNeeded (long)</td>
<td>D0: function result (long)</td>
</tr>
<tr>
<td>_ResrvMem</td>
<td>D0: cbNeeded (long)</td>
<td>D0: result code (word)</td>
</tr>
<tr>
<td>_PurgeMem</td>
<td>D0: cbNeeded (long)</td>
<td>D0: result code (word)</td>
</tr>
<tr>
<td>_EmptyHandle</td>
<td>A0: h (handle)</td>
<td>A0: h (handle)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>D0: result code (word)</td>
</tr>
<tr>
<td>_HLock</td>
<td>A0: h (handle)</td>
<td>D0: result code (word)</td>
</tr>
<tr>
<td>_HUnlock</td>
<td>A0: h (handle)</td>
<td>D0: result code (word)</td>
</tr>
<tr>
<td>_HPurge</td>
<td>A0: h (handle)</td>
<td>D0: result code (word)</td>
</tr>
<tr>
<td>_HNoPurge</td>
<td>A0: h (handle)</td>
<td>D0: result code (word)</td>
</tr>
<tr>
<td>_SetGrowZone</td>
<td>A0: growZone (ptr)</td>
<td>D0: result code (word)</td>
</tr>
<tr>
<td>_BlockMove</td>
<td>A0: sourcePtr (ptr)</td>
<td>D0: result code (word)</td>
</tr>
<tr>
<td></td>
<td>A1: destPtr (ptr)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>D0: byteCount (long)</td>
<td></td>
</tr>
</tbody>
</table>

**III-128 Memory Manager**
### Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DefltStack</td>
<td>Default space allotment for stack (long)</td>
</tr>
<tr>
<td>MinStack</td>
<td>Minimum space allotment for stack (long)</td>
</tr>
<tr>
<td>MemTop</td>
<td>Address of end of RAM (on Macintosh XL, end of RAM available to applications)</td>
</tr>
<tr>
<td>ScrnBase</td>
<td>Address of main screen buffer</td>
</tr>
<tr>
<td>BuffPtr</td>
<td>Address of end of jump table</td>
</tr>
<tr>
<td>CurrentA5</td>
<td>Address of boundary between application globals and application parameters</td>
</tr>
<tr>
<td>CurStackBase</td>
<td>Address of base of stack; start of application globals</td>
</tr>
<tr>
<td>ApplLimit</td>
<td>Application heap limit</td>
</tr>
<tr>
<td>HeapEnd</td>
<td>Address of end of application heap zone</td>
</tr>
<tr>
<td>ApplZone</td>
<td>Address of application heap zone</td>
</tr>
<tr>
<td>SysZone</td>
<td>Address of system heap zone</td>
</tr>
<tr>
<td>TheZone</td>
<td>Address of current heap zone</td>
</tr>
<tr>
<td>GZRootHnd</td>
<td>Handle to relocatable block not to be moved by grow zone function</td>
</tr>
</tbody>
</table>
MENU MANAGER

Constants

CONST { Value indicating item has no mark }

noMark = 0;

{ Messages to menu definition procedure }

mDrawMsg = 0; {draw the menu}
mChooseMsg = 1; {tell which item was chosen and highlight it}
mSizeMsg = 2; {calculate the menu's dimensions}

{ Resource ID of standard menu definition procedure }
textMenuProc = 0;

Data Types

TYPE MenuHandle = ^MenuPtr;
    MenuPtr = ^MenuInfo;
    MenuInfo = RECORD
        menuID: INTEGER; {menu ID}
        menuWidth: INTEGER; {menu width in pixels}
        menuHeight: INTEGER; {menu height in pixels}
        menuProc: Handle; {menu definition procedure}
        enableFlags: LONGINT; {tells if menu or items are enabled}
        menuData: Str255 {menu title (and other data)}
    END;

Routines

Initialization and Allocation

PROCEDURE InitMenus;
FUNCTION NewMenu (menuID: INTEGER; menuTitle: Str255) : MenuHandle;
FUNCTION GetMenu (resourceID: INTEGER) : MenuHandle;
PROCEDURE DisposeMenu (theMenu: MenuHandle);

Forming the Menus

PROCEDURE AppendMenu (theMenu: MenuHandle; data: Str255);
PROCEDURE AddResMenu (theMenu: MenuHandle; theType: ResType);
PROCEDURE InsertResMenu (theMenu: MenuHandle; theType: ResType; afterItem: INTEGER);
Forming the Menu Bar

PROCEDURE InsertMenu (theMenu: MenuHandle; beforeID: INTEGER);
PROCEDURE DrawMenuBar;
PROCEDURE DeleteMenu (menuID: INTEGER);
PROCEDURE ClearMenuBar;
FUNCTION GetNewMBar (menuBarID: INTEGER): Handle;
FUNCTION GetMenuBar: Handle;
PROCEDURE SetMenuBar (menuList: Handle);

Choosing From a Menu

FUNCTION MenuSelect (startPt: Point): LONGINT;
FUNCTION MenuKey (ch: CHAR): LONGINT;
PROCEDURE HiliteMenu (menuID: INTEGER);

Controlling the Appearance of Items

PROCEDURE SetItem (theMenu: MenuHandle; item: INTEGER; itemString: Str255);
PROCEDURE GetItem (theMenu: MenuHandle; item: INTEGER; VAR itemString: Str255);
PROCEDURE DisableItem (theMenu: MenuHandle; item: INTEGER);
PROCEDURE EnableItem (theMenu: MenuHandle; item: INTEGER);
PROCEDURE CheckItem (theMenu: MenuHandle; item: INTEGER; checked: BOOLEAN);
PROCEDURE SetItemMark (theMenu: MenuHandle; item: INTEGER; markChar: CHAR);
PROCEDURE GetItemMark (theMenu: MenuHandle; item: INTEGER; VAR markChar: CHAR);
PROCEDURE SetItemIcon (theMenu: MenuHandle; item: INTEGER; icon: Byte);
PROCEDURE GetItemIcon (theMenu: MenuHandle; item: INTEGER; VAR icon: Byte);
PROCEDURE SetItemStyle (theMenu: MenuHandle; item: INTEGER; chStyle: Style);
PROCEDURE GetItemStyle (theMenu: MenuHandle; item: INTEGER; VAR chStyle: Style);

Miscellaneous Routines

PROCEDURE CalcMenuSize (theMenu: MenuHandle);
FUNCTION CountMItems (theMenu: MenuHandle): INTEGER;
FUNCTION GetMHandle (menuID: INTEGER): MenuHandle;
PROCEDURE FlashMenuBar (menuID: INTEGER);
PROCEDURE SetMenuFlash (count: INTEGER);
Inside Macintosh

Meta-Characters for AppendMenu

<table>
<thead>
<tr>
<th>Meta-character or Return</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>; or Return</td>
<td>Separates multiple items</td>
</tr>
<tr>
<td>^</td>
<td>Followed by an icon number, adds that icon to the item</td>
</tr>
<tr>
<td>!</td>
<td>Followed by a character, marks the item with that character</td>
</tr>
<tr>
<td>&lt;</td>
<td>Followed by B, I, U, O, or S, sets the character style of the item</td>
</tr>
<tr>
<td>/</td>
<td>Followed by a character, associates a keyboard equivalent with the item</td>
</tr>
<tr>
<td>(</td>
<td>Disables the item</td>
</tr>
</tbody>
</table>

Menu Definition Procedure

PROCEDURE MyMenu (message: INTEGER; theMenu: MenuHandle; VAR menuRect: Rect; hitPt: Point; VAR whichItem: INTEGER);

Assembly-Language Information

Constants

; Value indicating item has no mark

noMark .EQU 0

; Messages to menu definition procedure

mDrawMsg .EQU 0 ; draw the menu
mChooseMsg .EQU 1 ; tell which item was chosen and highlight it
mSizeMsg .EQU 2 ; calculate the menu's dimensions

; Resource ID of standard menu definition procedure

textMenuProc .EQU 0

Menu Record Data Structure

- menuID: Menu ID (word)
- menuWidth: Menu width in pixels (word)
- menuHeight: Menu height in pixels (word)
- menuDefHandle: Handle to menu definition procedure
- menuEnable: Enable flags (long)
- menuData: Menu title (preceded by length byte) followed by data defining the items
- menuBlkSize: Size in bytes of menu record except menuData field

III-132 Menu Manager
### Special Macro Names

<table>
<thead>
<tr>
<th>Pascal name</th>
<th>Macro name</th>
</tr>
</thead>
<tbody>
<tr>
<td>DisposeMenu</td>
<td>_DisposMenu</td>
</tr>
<tr>
<td>GetItemIcon</td>
<td>_GetItmIcon</td>
</tr>
<tr>
<td>GetItemMark</td>
<td>_GetItmMark</td>
</tr>
<tr>
<td>GetItemStyle</td>
<td>_GetItmStyle</td>
</tr>
<tr>
<td>GetMenu</td>
<td>_GetRMenu</td>
</tr>
<tr>
<td>SetItemIcon</td>
<td>_SetItmIcon</td>
</tr>
<tr>
<td>SetItemMark</td>
<td>_SetItmMark</td>
</tr>
<tr>
<td>SetItemStyle</td>
<td>_SetItmStyle</td>
</tr>
<tr>
<td>SetMenuFlash</td>
<td>_SetMFlash</td>
</tr>
</tbody>
</table>

### Variables

- **MenuList**: Handle to current menu list
- **MBarEnable**: Nonzero if menu bar belongs to a desk accessory (word)
- **MenuHook**: Address of routine called repeatedly during MenuSelect
- **MBarHook**: Address of routine called by MenuSelect before menu is drawn (see below)
- **TheMenu**: Menu ID of currently highlighted menu (word)
- **MenuFlash**: Count for duration of menu item blinking (word)

**MBarHook routine**

- **On entry**: stack: pointer to menu rectangle
- **On exit**:
  - D0: 0 to continue MenuSelect
  - 1 to abort MenuSelect
PACKAGE MANAGER

Constants

CONST { Resource IDs for packages }

\[
\begin{align*}
dskInit = 2; & \quad \text{(Disk Initialization)} \\
stdFile = 3; & \quad \text{(Standard File)} \\
flPoint = 4; & \quad \text{(Floating-Point Arithmetic)} \\
trFunc = 5; & \quad \text{(Transcendental Functions)} \\
intUtil = 6; & \quad \text{(International Utilities)} \\
bdConv = 7; & \quad \text{(Binary-Decimal Conversion)}
\end{align*}
\]

Routines

PROEDURE InitPack (packID: INTEGER);
PROEDURE InitAllPacks;

Assembly-Language Information

Constants

; Resource IDs for packages

\[
\begin{align*}
dskInit & .EQU 2 \quad \text{;Disk Initialization} \\
stdFile & .EQU 3 \quad \text{;Standard File} \\
flPoint & .EQU 4 \quad \text{;Floating-Point Arithmetic} \\
trFunc & .EQU 5 \quad \text{;Transcendental Functions} \\
intUtil & .EQU 6 \quad \text{;International Utilities} \\
bdConv & .EQU 7 \quad \text{;Binary-Decimal Conversion}
\end{align*}
\]

Trap Macros for Packages

\[
\begin{align*}
\text{Disk Initialization} & \quad \_\text{Pack2} \\
\text{Standard File} & \quad \_\text{Pack3} \\
\text{Floating-Point Arithmetic} & \quad \_\text{Pack4} \quad (\text{synonym: } \_\text{FP68K}) \\
\text{Transcendental Functions} & \quad \_\text{Pack5} \quad (\text{synonym: } \_\text{Elems68K}) \\
\text{International Utilities} & \quad \_\text{Pack6} \\
\text{Binary-Decimal Conversion} & \quad \_\text{Pack7}
\end{align*}
\]
PRINTING MANAGER

Constants

CONST { Printing methods }

bDraftLoop = 0;  {draft printing}
bSpoolLoop = 1;  {spool printing}

{ Printer specification in prStl field of print record }
bDevCitoh = 1;  {Imagewriter printer}
bDevLaser = 3;  {LaserWriter printer}

{ Maximum number of pages in a spool file }
iPFMaxPgs = 128;

{ Result codes }
noErr  = 0;  {no error}
iPrSavPFil = -1;  {saving spool file}
controlErr = -17;  {unimplemented control instruction}
iIOAbort = -27;  {I/O abort error}
iMemFullErr = -108;  {not enough room in heap zone}
iPrAbort = 128;  {application or user requested abort}

{ PrCtlCall parameters }
iPrDevCtl = 7;  {printer control}
lPrReset = $00010000;  {reset printer}
lPrLineFeed = $00030000;  {carriage return only}
lPrLFSixth = $0003FFFF;  {standard 1/6-inch line feed}
lPrPageEnd = $00020000;  {end page}
iPrBitsCtl = 4;  {bit map printing}
lScreenBits = 0;  {default for printer}
lPaintBits = 1;  {square dots (72 by 72)}
iPrIOCtl = 5;  {text streaming}

{ Printer Driver information }
sPrDrvr = '.Print';  {Printer Driver resource name}
iPrDrvrRef = -3;  {Printer Driver reference number}
Data Types

TYPE TPPrPort = ^TPrPort;
TPrPort = RECORD
  gPort: GrafPort; {grafPort to draw in}
  {more fields for internal use}
END;

THPrint = ^TPPrint;
TPPrint = ^TPrint;
TPrint = RECORD
  iPrVersion: INTEGER; {Printing Manager version}
  prInfo: TPrInfo; {printer information subrecord}
  rPaper: Rect; {paper rectangle}
  prStl: TPrStl; {additional device information}
  prInfoPT: TPrInfo; {used internally}
  prXInfo: TPrXInfo; {additional device information}
  prJob: TPrJob; {job subrecord}
  printX: ARRAY[1..19] OF INTEGER {not used}
END;

TPrInfo = RECORD
  iDev: INTEGER; {used internally}
  iVRes: INTEGER; {vertical resolution of printer}
  iHRes: INTEGER; {horizontal resolution of printer}
  rPage: Rect {page rectangle}
END;

TPrJob = RECORD
  iFstPage: INTEGER; {first page to print}
  iLstPage: INTEGER; {last page to print}
  iCopies: INTEGER; {number of copies}
  bJDoeLoop: SignedByte; {printing method}
  fFromUsr: BOOLEAN; {used internally}
  pIdleProc: ProcPtr; {background procedure}
  pFileName: StringPtr; {spool file name}
  iFileVol: INTEGER; {spool file volume reference number}
  bFileVers: SignedByte; {spool file version number}
  bJobX: SignedByte {used internally}
END;

TPrStl = RECORD
  wDev: INTEGER; {high byte specifies device}
  {more fields for internal use}
END;
TPrXInfo = RECORD
  iRowBytes: INTEGER; {used internally}
iBandV: INTEGER; {used internally}
iBandH: INTEGER; {used internally}
iDevBytes: INTEGER; {size of buffer}
{more fields for internal use}
END;

TPRect = ^Rect;

TPrStatus = RECORD
  iTotPages: INTEGER; {number of pages in spool file}
iCurPage: INTEGER; {page being printed}
iTotCopies: INTEGER; {number of copies requested}
iCurCopy: INTEGER; {copy being printed}
iTotBands: INTEGER; {used internally}
iCurBand: INTEGER; {used internally}
fPgDirty: BOOLEAN; {TRUE if started printing page}
fImaging: BOOLEAN; {used internally}
hPrint: THPrint; {print record}
pPrPort: TPPrPort; {printing grafPort}
hPic: PicHandle {used internally}
END;

Routines [Not in ROM]

Initialization and Termination

PROCEDURE PrOpen;
PROCEDURE PrClose;

Print Records and Dialogs

PROCEDURE PrintDefault (hPrint: THPrint);
FUNCTION PrValidate (hPrint: THPrint) : BOOLEAN;
FUNCTION PrStdDialog (hPrint: THPrint) : BOOLEAN;
FUNCTION PrJobDialog (hPrint: THPrint) : BOOLEAN;
PROCEDURE PrJobMerge (hPrintSrc, hPrintDst: THPrint);

Printing

FUNCTION PrOpenDoc (hPrint: THPrint; pPrPort: TPPrPort; pIOBuf: Ptr) :
  TPPrPort;
PROCEDURE PrOpenPage (pPrPort: TPPrPort; pPageFrame: TPRect);
PROCEDURE PrClosePage (pPrPort: TPPrPort);
PROCEDURE PrCloseDoc (pPrPort: TPPrPort);
PROCEDURE PrPicFile (hPrint: THPrint; pPrPort: TPPrPort; pIOBuf: Ptr;
pDevBuf: Ptr; VAR prStatus: TPrStatus);
Error Handling

FUNCTION PrError : INTEGER;
PROCEDURE PrSetError (iErr: INTEGER);

Low-Level Driver Access

PROCEDURE PrDrvOpen;
PROCEDURE PrDrvClose;
PROCEDURE PrCtlCall (iWhichCtl: INTEGER; lParam1, lParam2, lParam3:
LONGLONG);
FUNCTION PrDrvDCE : Handle;
FUNCTION PrDrvVers : INTEGER;

Assembly-Language Information

Constants

; Printing methods
bDraftLoop .EQU 0 ;draft printing
bSpoolLoop .EQU 1 ;spool printing

; Result codes
noErr .EQU 0 ;no error
iPrSavPFil .EQU -1 ;saving spool file
cntolErr .EQU -17 ;unimplemented control instruction
iIOAbort .EQU -27 ;I/O abort error
iMemFullErr .EQU -108 ;not enough room in heap zone
iPrAbort .EQU 128 ;application or user requested abort

; Printer Driver Control call parameters
iPrDevCtl .EQU 7 ;printer control
lPrReset .EQU 1 ;reset printer
iPrLineFeed .EQU 3 ;carriage return/paper advance
iPrLFSixth .EQU 3 ;standard 1/6-inch line feed
lPrPageEnd .EQU 2 ;end page
iPrBitsCtl .EQU 4 ;bit map printing
lScreenBits .EQU 0 ;default for printer
lPaintBits .EQU 1 ;square dots (72 by 72)
iPrIOctl .EQU 5 ;text streaming

; Printer Driver information
iPrDrvRef .EQU -3 ;Printer Driver reference number

III-138 Printing Manager
Printing GrafPort Data Structure

gPort  GraFPort to draw in (portRec bytes)
iPrPortSize  Size in bytes of printing graFPort

Print Record Data Structure

iPrVersion  Printing Manager version (word)
prInfo  Printer information subrecord (14 bytes)
rPaper  Paper rectangle (8 bytes)
prStl  Additional device information (8 bytes)
prXInfo  Additional device information (16 bytes)
prJob  Job subrecord (iPrJobSize bytes)
iPrintSize  Size in bytes of print record

Structure of Printer Information Subrecord

iVRes  Vertical resolution of printer (word)
iHRes  Horizontal resolution of printer (word)
rPage  Page rectangle (8 bytes)

Structure of Job Subrecord

iFstPage  First page to print (word)
iLstPage  Last page to print (word)
iCopies  Number of copies (word)
bJDocLoop  Printing method (byte)
plidleProc  Address of background procedure
pFileName  Pointer to spool file name (preceded by length byte)
iFileVol  Spool file volume reference number (word)
bFileVers  Spool file version number (byte)
iPrJobSize  Size in bytes of job subrecord

Structure of PrXInfo Subrecord

iDevBytes  Size of buffer (word)

Structure of Printer Status Record

iTotPages  Number of pages in spool file (word)
iCurPage  Page being printed (word)
 iTotCopies  Number of copies requested (word)
iCurCopy  Copy being printed (word)
fPgDirty  Nonzero if started printing page (byte)
hPrint  Handle to print record
pPrPort  Pointer to printing graFPort
iPrStatSize  Size in bytes of printer status record
Variables

PrintErr: Result code from last Printing Manager routine (word)
CONST { Source transfer modes }

srcCopy = 0;
srcOr = 1;
srcXor = 2;
srcBic = 3;
notSrcCopy = 4;
notSrcOr = 5;
notSrcXor = 6;
notSrcBic = 7;

{ Pattern transfer modes }

patCopy = 8;
patOr = 9;
patXor = 10;
patBic = 11;
notPatCopy = 12;
notPatOr = 13;
notPatXor = 14;
notPatBic = 15;

{ Standard colors for ForeColor and BackColor }

blackColor = 33;
whiteColor = 30;
redColor = 205;
greenColor = 341;
blueColor = 409;
cyanColor = 273;
magentaColor = 137;
yellowColor = 69;

{ Standard picture comments }

picLParen = 0;
picRParen = 1;

TYPE StyleItem = (bold, italic, underline, outline, shadow, condense, extend);
Style = SET OF StyleItem;
VHSelect = (v, h);
Point = RECORD CASE INTEGER OF
  0: (v: INTEGER; {vertical coordinate}
      h: INTEGER); {horizontal coordinate}
  1: (vh: ARRAY[VHSelect] OF INTEGER)
END;

Rect = RECORD CASE INTEGER OF
  0: (top: INTEGER;
      left: INTEGER;
      bottom: INTEGER;
      right: INTEGER);
  1: (topLeft: Point;
      botRight: Point)
END;

RgnHandle = *RgnPtr;
RgnPtr = *Region;
Region = RECORD
  rgnSize: INTEGER; {size in bytes}
  rgnBBox: Rect; {enclosing rectangle}
  {more data if not rectangular}
END;

BitMap = RECORD
  baseAddr: Ptr; {pointer to bit image}
  rowBytes: INTEGER; {row width}
  bounds: Rect {boundary rectangle}
END;

Pattern = PACKED ARRAY[0..7] OF 0..255;

Bits16 = ARRAY[0..15] OF INTEGER;

Cursor = RECORD
  data: Bits16; {cursor image}
  mask: Bits16; {cursor mask}
  hotspot: Point {point aligned with mouse}
END;
QDProcsPtr = ^QDProcs;
QDProcs = RECORD
  textProc: Ptr; {text drawing}
  lineProc: Ptr; {line drawing}
  rectProc: Ptr; {rectangle drawing}
  rRectProc: Ptr; {roundRect drawing}
  ovalProc: Ptr; {oval drawing}
  arcProc: Ptr; {arc/wedge drawing}
  rgnProc: Ptr; {region drawing}
  bitsProc: Ptr; {bit transfer}
  commentProc: Ptr; {picture comment processing}
  txMeasProc: Ptr; {text width measurement}
  getPicProc: Ptr; {picture retrieval}
  putPicProc: Ptr; {picture saving}
END;

GrafPtr = ^GrafPort;
GrafPort = RECORD
  device: INTEGER; {device-specific information}
  portBits: BitMap; {grafPort's bit map}
  portRect: Rect; {grafPort's rectangle}
  visRgn: RgnHandle; {visible region}
  clipRgn: RgnHandle; {clipping region}
  bkPat: Pattern; {background pattern}
  fillPat: Pattern; {fill pattern}
  pnLoc: Point; {pen location}
  pnSize: Point; {pen size}
  pnMode: INTEGER; {pen's transfer mode}
  pnPat: Pattern; {pen pattern}
  pnVis: INTEGER; {pen visibility}
  txFont: INTEGER; {font number for text}
  txFace: Style; {text's character style}
  txMode: INTEGER; {text's transfer mode}
  txSize: INTEGER; {font size for text}
  spExtra: Fixed; {extra space}
  fgColor: LONGINT; {foreground color}
  bkColor: LONGINT; {background color}
  colrBit: INTEGER; {color bit}
  patStretch: INTEGER; {used internally}
  picSave: Handle; {picture being saved}
  rgnSave: Handle; {region being saved}
  polySave: Handle; {polygon being saved}
  grafProcs: QDProcsPtr {low-level drawing routines}
END;

PicHandle = ^PicPtr;
PicPtr = ^Picture;
Picture = RECORD
  picSize: INTEGER; {size in bytes}
  picFrame: Rect; {picture frame}
END;
PolyHandle = PolyPtr;
PolyPtr = Polygon;
Polygon = RECORD
  polySize: INTEGER; {size in bytes}
  polyBBox: Rect; {enclosing rectangle}
  polyPoints: ARRAY[0..0] OF Point
END;

PenState = RECORD
  pnLoc: Point; {pen location}
  pnSize: Point; {pen size}
  pnMode: INTEGER; {pen's transfer mode}
  pnPat: Pattern {pen pattern}
END;

FontInfo = RECORD
  ascent: INTEGER; {ascent}
  descent: INTEGER; {descent}
  widMax: INTEGER; {maximum character width}
  leading: INTEGER {leading}
END;

GrafVerb = (frame, paint, erase, invert, fill);

Variables

VAR thePort: GrafPtr; {pointer to current grafPort}
  white: Pattern; {all-white pattern}
  black: Pattern; {all-black pattern}
  gray: Pattern; {50% gray pattern}
  ltGray: Pattern; {25% gray pattern}
  dkGray: Pattern; {75% gray pattern}
  arrow: Cursor; {standard arrow cursor}
  screenBits: Bitmap; {the entire screen}
  randSeed: LONGINT; {determines where Random sequence begins}

Routines

GrafPort Routines

PROCEDURE InitGraf (globalPtr: Ptr);
PROCEDURE OpenPort (port: GrafPtr);
PROCEDURE InitPort (port: GrafPtr);
PROCEDURE ClosePort (port: GrafPtr);
PROCEDURE SetPort (port: GrafPtr);
PROCEDURE GetPort (VAR port: GrafPtr);
PROCEDURE GrafDevice (device: INTEGER);
PROCEDURE SetPortBits (bm: Bitmap);
PROCEDURE PortSize (width, height: INTEGER);
PROCEDURE MovePortTo (leftGlobal, topGlobal: INTEGER);
PROCEDURE SetOrigin (h, v: INTEGER);
PROCEDURE SetClip (rgn: RgnHandle);
PROCEDURE GetClip (rgn: RgnHandle);
PROCEDURE ClipRect (r: Rect);
PROCEDURE BackPat (pat: Pattern);

**Cursor Handling**

PROCEDURE InitCursor;
PROCEDURE SetCursor (crsr: Cursor);
PROCEDURE HideCursor;
PROCEDURE ShowCursor;
PROCEDURE ObscureCursor;

**Pen and Line Drawing**

PROCEDURE HidePen;
PROCEDURE ShowPen;
PROCEDURE GetPen (VAR pt: Point);
PROCEDURE GetPenState (VAR pnState: PenState);
PROCEDURE SetPenState (pnState: PenState);
PROCEDURE PenSize (width, height: INTEGER);
PROCEDURE PenMode (mode: INTEGER);
PROCEDURE PenPat (pat: Pattern);
PROCEDURE PenNormal;
PROCEDURE MoveTo (h, v: INTEGER);
PROCEDURE Move (dh, dv: INTEGER);
PROCEDURE LineTo (h, v: INTEGER);
PROCEDURE Line (dh, dv: INTEGER);

**Text Drawing**

PROCEDURE TextFont (font: INTEGER);
PROCEDURE TextFace (face: Style);
PROCEDURE TextMode (mode: INTEGER);
PROCEDURE TextSize (size: INTEGER);
PROCEDURE SpaceExtra (extra: Fixed);
PROCEDURE DrawChar (ch: CHAR);
PROCEDURE DrawString (s: Str255);
PROCEDURE DrawText (textBuf: Ptr; firstByte, byteCount: INTEGER);
FUNCTION CharWidth (ch: CHAR): INTEGER;
FUNCTION StringWidth (s: Str255): INTEGER;
FUNCTION TextWidth (textBuf: Ptr; firstByte, byteCount: INTEGER):
  INTEGER;
PROCEDURE GetFontInfo (VAR info: FontInfo);
Drawing in Color

PROCEDURE ForeColor (color: LONGINT);
PROCEDURE BackColor (color: LONGINT);
PROCEDURE ColorBit (whichBit: INTEGER);

Calculations with Rectangles

PROCEDURE SetRect (VAR r: Rect; left, top, right, bottom: INTEGER);
PROCEDURE OffsetRect (VAR r: Rect; dh, dv: INTEGER);
PROCEDURE InsetRect (VAR r: Rect; dh, dv: INTEGER);
FUNCTION SectRect (src1, src2: Rect; VAR dstRect: Rect): BOOLEAN;
PROCEDURE UnionRect (src1, src2: Rect; VAR dstRect: Rect);
FUNCTION PtInRect (pt: Point; r: Rect): BOOLEAN;
PROCEDURE Pt2Rect (pt1, pt2: Point; VAR dstRect: Rect);
FUNCTION PtToAngle (r: Rect; pt: Point; VAR angle: INTEGER);
FUNCTION EqualRect (rect1, rect2: Rect): BOOLEAN;
FUNCTION EmptyRect (r: Rect): BOOLEAN;

Graphic Operations on Rectangles

PROCEDURE FrameRect (r: Rect);
PROCEDURE PaintRect (r: Rect);
PROCEDURE EraseRect (r: Rect);
PROCEDURE InvertRect (r: Rect);
PROCEDURE FillRect (r: Rect; pat: Pattern);

Graphic Operations on Ovals

PROCEDURE FrameOval (r: Rect);
PROCEDURE PaintOval (r: Rect);
PROCEDURE EraseOval (r: Rect);
PROCEDURE InvertOval (r: Rect);
PROCEDURE FillOval (r: Rect; pat: Pattern);

Graphic Operations on Rounded-Corner Rectangles

PROCEDURE FrameRoundRect (r: Rect; ovalWidth, ovalHeight: INTEGER);
PROCEDURE PaintRoundRect (r: Rect; ovalWidth, ovalHeight: INTEGER);
PROCEDURE EraseRoundRect (r: Rect; ovalWidth, ovalHeight: INTEGER);
PROCEDURE InvertRoundRect (r: Rect; ovalWidth, ovalHeight: INTEGER);
PROCEDURE FillRoundRect (r: Rect; ovalWidth, ovalHeight: INTEGER; pat: Pattern);
Graphic Operations on Arcs and Wedges

PROCEDURE FrameArc  \( (r: \text{Rect}; \text{startAngle,arcAngle}: \text{INTEGER}) \);
PROCEDURE PaintArc   \( (r: \text{Rect}; \text{startAngle,arcAngle}: \text{INTEGER}) \);
PROCEDURE EraseArc   \( (r: \text{Rect}; \text{startAngle,arcAngle}: \text{INTEGER}) \);
PROCEDURE InvertArc  \( (r: \text{Rect}; \text{startAngle,arcAngle}: \text{INTEGER}) \);
PROCEDURE FillArc    \( (r: \text{Rect}; \text{startAngle,arcAngle}: \text{INTEGER}; \text{pat}: \text{Pattern}) \);

Calculations with Regions

FUNCTION NewRgn     \( : \text{RgnHandle;} \)
PROCEDURE OpenRgn;
PROCEDURE CloseRgn   \( (\text{dstRgn}: \text{RgnHandle}) \);
PROCEDURE DisposeRgn \( (\text{rgn}: \text{RgnHandle}) \);
PROCEDURE CopyRgn    \( (\text{srcRgn, dstRgn}: \text{RgnHandle}) \);
PROCEDURE SetEmptyRgn\( (\text{rgn}: \text{RgnHandle}) \);
PROCEDURE SetRectRgn \( (\text{rgn}: \text{RgnHandle}; \text{left, top, right, bottom}: \text{INTEGER}) \);
PROCEDURE RectRgn    \( (\text{rgn}: \text{RgnHandle}; \text{r}: \text{Rect}) \);
PROCEDURE OffsetRgn  \( (\text{rgn}: \text{RgnHandle}; \text{dh, dv}: \text{INTEGER}) \);
PROCEDURE InsetRgn   \( (\text{rgn}: \text{RgnHandle}; \text{dh, dv}: \text{INTEGER}) \);
PROCEDURE SectRgn    \( (\text{srcRgnA,srcRgnB, dstRgn}: \text{RgnHandle}) \);
PROCEDURE UnionRgn   \( (\text{srcRgnA, srcRgnB, dstRgn}: \text{RgnHandle}) \);
PROCEDURE DiffRgn    \( (\text{srcRgnA, srcRgnB, dstRgn}: \text{RgnHandle}) \);
PROCEDURE XorRgn     \( (\text{srcRgnA, srcRgnB, dstRgn}: \text{RgnHandle}) \);
FUNCTION PtInRgn     \( (\text{rgn}: \text{RgnHandle}; \text{pt}: \text{Point}) : \text{BOOLEAN} \);
FUNCTION RectInRgn   \( (\text{r}: \text{Rect}; \text{rgn}: \text{RgnHandle}) : \text{BOOLEAN} \);
FUNCTION EqualRgn    \( (\text{rgnA, rgnB}: \text{RgnHandle}) : \text{BOOLEAN} \);
FUNCTION EmptyRgn    \( (\text{rgn}: \text{RgnHandle}) : \text{BOOLEAN} \);

Graphic Operations on Regions

PROCEDURE FrameRgn   \( (\text{rgn}: \text{RgnHandle}) \);
PROCEDURE PaintRgn   \( (\text{rgn}: \text{RgnHandle}) \);
PROCEDURE EraseRgn   \( (\text{rgn}: \text{RgnHandle}) \);
PROCEDURE InvertRgn  \( (\text{rgn}: \text{RgnHandle}) \);
PROCEDURE FillRgn    \( (\text{rgn}: \text{RgnHandle}; \text{pat}: \text{Pattern}) \);

Bit Transfer Operations

PROCEDURE ScrollRect \( (\text{r}: \text{Rect}; \text{dh, dv}: \text{INTEGER}; \text{updateRgn}: \text{RgnHandle}) \);
PROCEDURE CopyBits   \( (\text{srcBits, dstBits}: \text{BitMap}; \text{srcRect, dstRect}: \text{Rect} ; \text{mode}: \text{INTEGER}; \text{maskRgn}: \text{RgnHandle}) \);
Pictures

FUNCTION OpenPicture (picFrame: Rect): PicHandle;
PROCEDURE PicComment (kind, dataSize: INTEGER; dataHandle: Handle);
PROCEDURE ClosePicture;
PROCEDURE DrawPicture (myPicture: PicHandle; dstRect: Rect);
PROCEDURE KillPicture (myPicture: PicHandle);

Calculations with Polygons

FUNCTION OpenPoly: PolyHandle;
PROCEDURE ClosePoly;
PROCEDURE KillPoly (poly: PolyHandle);
PROCEDURE OffsetPoly (poly: PolyHandle; dh, dv: INTEGER);

Graphic Operations on Polygons

PROCEDURE FramePoly (poly: PolyHandle);
PROCEDURE PaintPoly (poly: PolyHandle);
PROCEDURE ErasePoly (poly: PolyHandle);
PROCEDURE InvertPoly (poly: PolyHandle);
PROCEDURE FillPoly (poly: PolyHandle; pat: Pattern);

Calculations with Points

PROCEDURE AddPt (srcPt: Point; VAR dstPt: Point);
PROCEDURE SubPt (srcPt: Point; VAR dstPt: Point);
PROCEDURE SetPt (VAR pt: Point; h, v: INTEGER);
FUNCTION EqualPt (pt1, pt2: Point): BOOLEAN;
PROCEDURE LocalToGlobal (VAR pt: Point);
PROCEDURE GlobalToLocal (VAR pt: Point);

Miscellaneous Routines

FUNCTION Random: INTEGER;
FUNCTION GetPixel (h, v: INTEGER): BOOLEAN;
PROCEDURE StuffHex (thingPtr: Ptr; s: Str255);
PROCEDURE ScalePt (VAR pt: Point; srcRect, dstRect: Rect);
PROCEDURE MapPt (VAR pt: Point; srcRect, dstRect: Rect);
PROCEDURE MapRect (VAR r: Rect; srcRect, dstRect: Rect);
PROCEDURE MapRgn (rgn: RgnHandle; srcRect, dstRect: Rect);
PROCEDURE MapPoly (poly: PolyHandle; srcRect, dstRect: Rect);
Customizing QuickDraw Operations

PROCEDURE SetStdProcs (VAR procs: QDProcs);
PROCEDURE StdText (byteCount: INTEGER; textBuf: Ptr; numer, denom: Point);
PROCEDURE StdLine (newPt: Point);
PROCEDURE StdRect (verb: GrafVerb; r: Rect);
PROCEDURE StdRRect (verb: GrafVerb; r: Rect; ovalwidth, ovalHeight: INTEGER);
PROCEDURE StdOval (verb: GrafVerb; r: Rect);
PROCEDURE StdArc (verb: GrafVerb; r: Rect; startAngle, arcAngle: INTEGER);
PROCEDURE StdPoly (verb: GrafVerb; poly: PolyHandle);
PROCEDURE StdRgn (verb: GrafVerb; rgn: RgnHandle);
PROCEDURE StdBits (VAR srcBits: BitMap; VAR srcRect, dstRect: Rect;
                  mode: INTEGER; maskRgn: RgnHandle);
PROCEDURE StdComment (kind, dataSize: INTEGER; dataHandle: Handle);
FUNCTION StdTxMeas (byteCount: INTEGER; textAddr: Ptr; VAR numer, denom: Point;
                     VAR info: FontInfo): INTEGER;
PROCEDURE StdGetPic (dataPtr: Ptr; byteCount: INTEGER);
PROCEDURE StdPutPic (dataPtr: Ptr; byteCount: INTEGER);

Assembly-Language Information

Constants

; Size in bytes of QuickDraw global variables

grafSize .EQU 206

; Source transfer modes

srcCopy .EQU 0
srcOr .EQU 1
srcXor .EQU 2
srcBic .EQU 3
notSrcCopy .EQU 4
notSrcOr .EQU 5
notSrcXor .EQU 6
notSrcBic .EQU 7

; Pattern transfer modes

patCopy .EQU 8
patOr .EQU 9
patXor .EQU 10
patBic .EQU 11
notPatCopy .EQU 12
notPatOr .EQU 13
notPatXor .EQU 14
notPatBic .EQU 15
Inside Macintosh

; Standard colors for ForeColor and BackColor

blackColor .EQU 33
whiteColor .EQU 30
redColor .EQU 205
greenColor .EQU 341
blueColor .EQU 409
cyanColor .EQU 273
magentaColor .EQU 137
yellowColor .EQU 69

; Standard picture comments

picLParen .EQU 0
picRParen .EQU 1

; Character style

boldBit .EQU 0
italicBit .EQU 1
ulineBit .EQU 2
outlineBit .EQU 3
shadowBit .EQU 4
condenseBit .EQU 5
extendBit .EQU 6

; Graphic operations

frame .EQU 0
paint .EQU 1
erase .EQU 2
invert .EQU 3
fill .EQU 4

Point Data Structure

v Vertical coordinate (word)
h Horizontal coordinate (word)

Rectangle Data Structure

top Vertical coordinate of top left corner (word)
left Horizontal coordinate of top left corner (word)
bottom Vertical coordinate of bottom right corner (word)
right Horizontal coordinate of bottom right corner (word)
topLeft Top left corner (point; long)
botRight Bottom right corner (point; long)

III-150 QuickDraw
Region Data Structure

- **rgnSize**: Size in bytes (word)
- **rgnBBox**: Enclosing rectangle (8 bytes)
- **rgnData**: More data if not rectangular

Bit Map Data Structure

- **baseAddr**: Pointer to bit image
- **rowBytes**: Row width (word)
- **bounds**: Boundary rectangle (8 bytes)
- **bitMapRec**: Size in bytes of bit map data structure

Cursor Data Structure

- **data**: Cursor image (32 bytes)
- **mask**: Cursor mask (32 bytes)
- **hotSpot**: Point aligned with mouse (long)
- **cursRec**: Size in bytes of cursor data structure

Structure of QDProcs Record

- **textProc**: Address of text-drawing routine
- **lineProc**: Address of line-drawing routine
- **rectProc**: Address of rectangle-drawing routine
- **rRectProc**: Address of roundRect-drawing routine
- **ovalProc**: Address of oval-drawing routine
- **arcProc**: Address of arc/wedge-drawing routine
- **polyProc**: Address of polygon-drawing routine
- **rgnProc**: Address of region-drawing routine
- **bitsProc**: Address of bit-transfer routine
- **commentProc**: Address of routine for processing picture comments
- **txMeasProc**: Address of routine for measuring text width
- **getPicProc**: Address of picture-retrieval routine
- **putPicProc**: Address of picture-saving routine
- **qdProcsRec**: Size in bytes of QDProcs record

GrafPort Data Structure

- **device**: Font-specific information (word)
- **portBits**: GrafPort's bit map (bitMapRec bytes)
- **portBounds**: Boundary rectangle of grafPort's bit map (8 bytes)
- **portRect**: GrafPort's rectangle (8 bytes)
- **visRgn**: Handle to visible region
- **clipRgn**: Handle to clipping region
- **bkPat**: Background pattern (8 bytes)
- **fillPat**: Fill pattern (8 bytes)
- **pnLoc**: Pen location (point; long)
Inside Macintosh

- `pnSize`: Pen size (point; long)
- `pnMode`: Pen's transfer mode (word)
- `pnPat`: Pen pattern (8 bytes)
- `pnVis`: Pen visibility (word)
- `txFont`: Font number for text (word)
- `txFace`: Text's character style (word)
- `txMode`: Text's transfer mode (word)
- `txSize`: Font size for text (word)
- `spExtra`: Extra space (long)
- `fgColor`: Foreground color (long)
- `bkColor`: Background color (long)
- `colrBit`: Color bit (word)
- `picSave`: Handle to picture being saved
- `rgnSave`: Handle to region being saved
- `polySave`: Handle to polygon being saved
- `grafProcs`: Pointer to QDProcs record
- `portRec`: Size in bytes of grafPort

**Picture Data Structure**

- `picSize`: Size in bytes (word)
- `picFrame`: Picture frame (rectangle; 8 bytes)
- `picData`: Picture definition data

**Polygon Data Structure**

- `polySize`: Size in bytes (word)
- `polyBBox`: Enclosing rectangle (8 bytes)
- `polyPoints`: Polygon points

**Pen State Data Structure**

- `psLoc`: Pen location (point; long)
- `psSize`: Pen size (point; long)
- `psMode`: Pen's transfer mode (word)
- `psPat`: Pen pattern (8 bytes)
- `psRec`: Size in bytes of pen state data structure

**Font Information Data Structure**

- `ascent`: Ascent (word)
- `descent`: Descent (word)
- `widMax`: Maximum character width (word)
- `leading`: Leading (word)
### Special Macro Names

<table>
<thead>
<tr>
<th>Pascal name</th>
<th>Macro name</th>
</tr>
</thead>
<tbody>
<tr>
<td>SetPortBits</td>
<td>_SetPBits</td>
</tr>
<tr>
<td>InvertRect</td>
<td>_InverRect</td>
</tr>
<tr>
<td>InvertRoundRect</td>
<td>_InverRoundRect</td>
</tr>
<tr>
<td>DisposeRgn</td>
<td>_DisposRgn</td>
</tr>
<tr>
<td>SetRectRgn</td>
<td>_SetRecRgn</td>
</tr>
<tr>
<td>OffsetRgn</td>
<td>_OfSetRgn</td>
</tr>
<tr>
<td>InvertRgn</td>
<td>_InverRgn</td>
</tr>
<tr>
<td>ClosePoly</td>
<td>_ClosePgon</td>
</tr>
</tbody>
</table>

### Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RndSeed</td>
<td>Random number seed (long)</td>
</tr>
</tbody>
</table>
RESOURCE MANAGER

Constants

CONST { Masks for resource attributes }
resSysHeap = 64; {set if read into system heap}
resPurgeable = 32; {set if purgeable}
resLocked = 16; {set if locked}
resProtected = 8; {set if protected}
resPreload = 4; {set if to be preloaded}
resChanged = 2; {set if to be written to resource file}

{ Resource Manager result codes }
resNotFound = -192; {resource not found}
resFNotFound = -193; {resource file not found}
addResFailed = -194; {AddResource failed}
rmvResFailed = -196; {RmveResource failed}

{ Masks for resource file attributes }
mapReadOnly = 128; {set if file is read-only}
mapCompact = 64; {set to compact file on update}
mapChanged = 32; {set to write map on update}

Data Types

TYPE ResType = PACKED ARRAY[1..4] OF CHAR;

Routines

Initialization

FUNCTION InitResources : INTEGER;
PROCEDURE RsrcZoneInit;

Opening and Closing Resource Files

PROCEDURE CreateResFile (fileName: Str255);
FUNCTION OpenResFile (fileName: Str255) : INTEGER;
PROCEDURE CloseResFile (refNum: INTEGER);
Checking for Errors

FUNCTION ResError : INTEGER;

Setting the Current Resource File

FUNCTION CurResFile : INTEGER;
FUNCTION HomeResFile (theResource: Handle) : INTEGER;
PROCEDURE UseResFile (refNum: INTEGER);

Getting Resource Types

FUNCTION CountTypes : INTEGER;
PROCEDURE GetIndType (VAR theType: ResType; index: INTEGER);

Getting and Disposing of Resources

PROCEDURE SetResLoad (load: BOOLEAN);
FUNCTION CountResources (theType: ResType) : INTEGER;
FUNCTION GetIndResource (theType: ResType; index: INTEGER) : Handle;
FUNCTION GetResource (theType: ResType; theID: INTEGER) : Handle;
FUNCTION GetNamedResource (theType: ResType; name: Str255) : Handle;
PROCEDURE LoadResource (theResource: Handle);
PROCEDURE ReleaseResource (theResource: Handle);
PROCEDURE DetachResource (theResource: Handle);

Getting Resource Information

FUNCTION UniqueID (theType: ResType) : INTEGER;
PROCEDURE GetResInfo (theResource: Handle; VAR theID: INTEGER; VAR theType: ResType; VAR name: Str255);
FUNCTION GetResAttrs (theResource: Handle) : INTEGER;
FUNCTION SizeResource (theResource: Handle) : LONGINT;

Modifying Resources

PROCEDURE SetResInfo (theResource: Handle; theID: INTEGER; name: Str255);
PROCEDURE SetResAttrs (theResource: Handle; attrs: INTEGER);
PROCEDURE ChangedResource (theResource: Handle);
PROCEDURE AddResource (theData: Handle; theType: ResType; theID: INTEGER; name: Str255);
PROCEDURE RemoveResource (theResource: Handle);
PROCEDURE UpdateResFile (refNum: INTEGER);
PROCEDURE WriteResource (theResource: Handle);
PROCEDURE SetResPurge (install: BOOLEAN);
Advanced Routines

FUNCTION GetResFileAttrs (refNum: INTEGER) : INTEGER;
PROCEDURE SetResFileAttrs (refNum: INTEGER; attrs: INTEGER);

Assembly-Language Information

Constants

; Resource attributes

resSysHeap .EQU 6 ; set if read into system heap
resPurgeable .EQU 5 ; set if purgeable
resLocked .EQU 4 ; set if locked
resProtected .EQU 3 ; set if protected
resPreload .EQU 2 ; set if to be preloaded
resChanged .EQU 1 ; set if to be written to resource file

; Resource Manager result codes

resNotFound .EQU -192 ; resource not found
resFNotFound .EQU -193 ; resource file not found
addResFailed .EQU -194 ; AddResource failed
rmvResFailed .EQU -196 ; RmveResource failed

; Resource file attributes

mapReadOnly .EQU 7 ; set if file is read-only
mapCompact .EQU 6 ; set to compact file on update
mapChanged .EQU 5 ; set to write map on update

Special Macro Names

Pascal name  
Macro name
SizeResource  _SizeRsrc

Variables

TopMapHndl   Handle to resource map of most recently opened resource file
SysMapHndl   Handle to map of system resource file
SysMap       Reference number of system resource file (word)
CurMap       Reference number of current resource file (word)
ResLoad      Current SetResLoad state (word)
ResErr       Current value of ResError (word)
ResErrProc   Address of resource error procedure
SysResName   Name of system resource file (length byte followed by up to 19 characters)
SCRAP MANAGER

Constants

CONST { Scrap Manager result codes }

noScrapErr = -100; {desk scrap isn't initialized}
noTypeErr = -102; {no data of the requested type}

Data Types

TYPE PScrapStuff = ^ScrapStuff;
ScrapStuff = RECORD
  scrapSize:  LONGINT; {size of desk scrap}
  scrapHandle: Handle; {handle to desk scrap}
  scrapCount:  INTEGER; {count changed by ZeroScrap}
  scrapState:  INTEGER; {tells where desk scrap is}
  scrapName:  StringPtr  {scrap file name}
END;

Routines

Getting Desk Scrap Information

FUNCTION InfoScrap : PScrapStuff;

Keeping the Desk Scrap on the Disk

FUNCTION UnloadScrap : LONGINT;
FUNCTION LoadScrap :  LONGINT;

Writing to the Desk Scrap

FUNCTION ZeroScrap : LONGINT;
FUNCTION PutScrap (length: LONGINT; theType: ResType; source: Ptr) :
  LONGINT;

Reading from the Desk Scrap

FUNCTION GetScrap (hDest: Handle; theType: ResType; VAR offset: LONGINT) :
  LONGINT;
Assembly-Language Information

Constants

; Scrap Manager result codes

noScrapErr  .EQU  -100  ;desk scrap isn't initialized
noTypeErr   .EQU  -102  ;no data of the requested type

Special Macro Names

<table>
<thead>
<tr>
<th>Pascal name</th>
<th>Macro name</th>
</tr>
</thead>
<tbody>
<tr>
<td>LoadScrap</td>
<td>_LodeScrap</td>
</tr>
<tr>
<td>UnloadScrap</td>
<td>_UnlodeScrap</td>
</tr>
</tbody>
</table>

Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ScrapSize</td>
<td>Size in bytes of desk scrap (long)</td>
</tr>
<tr>
<td>ScrapHandle</td>
<td>Handle to desk scrap in memory</td>
</tr>
<tr>
<td>ScrapCount</td>
<td>Count changed by ZeroScrap (word)</td>
</tr>
<tr>
<td>ScrapState</td>
<td>Tells where desk scrap is (word)</td>
</tr>
<tr>
<td>ScrapName</td>
<td>Pointer to scrap file name (preceded by length byte)</td>
</tr>
</tbody>
</table>

III-158 Scrap Manager
SEGMENT LOADER

Constants

CONST { Message returned by CountAppleFiles }

appOpen = 0; {open the document(s)}
appPrint = 1; {print the document(s)}

Data Types

TYPE AppFile = RECORD
  vRefNum: INTEGER; {volume reference number}
fType: OSType; {file type}
versNum: INTEGER; {version number}
fName: String 255 {file name}
END;

Routines

PROCEDURE CountAppFiles (VAR message: INTEGER; VAR count: INTEGER); [Not in ROM]
PROCEDURE GetAppFiles (index: INTEGER; VAR theFile: AppFile); [Not in ROM]
PROCEDURE ClrAppFiles (index: INTEGER); [Not in ROM]
PROCEDURE GetAppParms (VAR apName: String 255; VAR apRefNum: INTEGER; VAR apParam: Handle);
PROCEDURE UnloadSeg (routineAddr: Ptr);
PROCEDURE ExitToShell;

Assembly-Language Information

Advanced Routines

Trap macro  On entry
  _Chain  (A0): pointer to application's file name (preceded by length byte)
            4(A0): configuration of sound and screen buffers (word)
  _Launch (A0): pointer to application's file name (preceded by length byte)
            4(A0): configuration of sound and screen buffers (word)
  _LoadSeg stack: segment number (word)
**Variables**

- **AppParmHandle**: Handle to Finder information
- **CurApName**: Name of current application (length byte followed by up to 31 characters)
- **CurApRefNum**: Reference number of current application's resource file (word)
- **CurPageOption**: Sound/screen buffer configuration passed to Chain or Launch (word)
- **CurJTOffset**: Offset to jump table from location pointed to by A5 (word)
- **FinderName**: Name of the Finder (length byte followed by up to 15 characters)
# SERIAL DRIVERS

## Constants

```
CONST { Driver reset information }

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>baud300</td>
<td>380;</td>
<td>(300 baud)</td>
</tr>
<tr>
<td>baud600</td>
<td>189;</td>
<td>(600 baud)</td>
</tr>
<tr>
<td>baud1200</td>
<td>94;</td>
<td>(1200 baud)</td>
</tr>
<tr>
<td>baud1800</td>
<td>62;</td>
<td>(1800 baud)</td>
</tr>
<tr>
<td>baud2400</td>
<td>46;</td>
<td>(2400 baud)</td>
</tr>
<tr>
<td>baud3600</td>
<td>30;</td>
<td>(3600 baud)</td>
</tr>
<tr>
<td>baud4800</td>
<td>22;</td>
<td>(4800 baud)</td>
</tr>
<tr>
<td>baud7200</td>
<td>14;</td>
<td>(7200 baud)</td>
</tr>
<tr>
<td>baud9600</td>
<td>10;</td>
<td>(9600 baud)</td>
</tr>
<tr>
<td>baud19200</td>
<td>4;</td>
<td>(19200 baud)</td>
</tr>
<tr>
<td>baud57600</td>
<td>0;</td>
<td>(57600 baud)</td>
</tr>
<tr>
<td>stop10</td>
<td>16384;</td>
<td>(1 stop bit)</td>
</tr>
<tr>
<td>stop15</td>
<td>-32768;</td>
<td>(1.5 stop bits)</td>
</tr>
<tr>
<td>stop20</td>
<td>-16384;</td>
<td>(2 stop bits)</td>
</tr>
<tr>
<td>noParity</td>
<td>0;</td>
<td>(no parity)</td>
</tr>
<tr>
<td>oddParity</td>
<td>4096;</td>
<td>(odd parity)</td>
</tr>
<tr>
<td>evenParity</td>
<td>12288;</td>
<td>(even parity)</td>
</tr>
<tr>
<td>data5</td>
<td>0;</td>
<td>(5 data bits)</td>
</tr>
<tr>
<td>data6</td>
<td>2048;</td>
<td>(6 data bits)</td>
</tr>
<tr>
<td>data7</td>
<td>1024;</td>
<td>(7 data bits)</td>
</tr>
<tr>
<td>data8</td>
<td>3072;</td>
<td>(8 data bits)</td>
</tr>
</tbody>
</table>

{ Masks for errors }

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>swOverrunErr</td>
<td>1;</td>
<td>(set if software overrun error)</td>
</tr>
<tr>
<td>parityErr</td>
<td>16;</td>
<td>(set if parity error)</td>
</tr>
<tr>
<td>hwOverrunErr</td>
<td>32;</td>
<td>(set if hardware overrun error)</td>
</tr>
<tr>
<td>framingErr</td>
<td>64;</td>
<td>(set if framing error)</td>
</tr>
</tbody>
</table>

{ Masks for changes that cause events to be posted }

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ctsEvent</td>
<td>32;</td>
<td>(set if CTS change will cause event to be posted)</td>
</tr>
<tr>
<td>breakEvent</td>
<td>128;</td>
<td>(set if break status change will cause event to be posted)</td>
</tr>
</tbody>
</table>

{ Indication that an XOff character was sent }

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>xOffWasSent</td>
<td>$80;</td>
<td></td>
</tr>
</tbody>
</table>

{ Result codes }

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>noErr</td>
<td>0;</td>
<td>(no error)</td>
</tr>
<tr>
<td>openErr</td>
<td>-23;</td>
<td>(attempt to open RAM Serial Driver failed)</td>
</tr>
</tbody>
</table>
```
Data Types

TYPE SPortSel = (sPortA, {modem port}
  sPortB {printer port});

SerShk = PACKED RECORD
  fXOn: Byte; {XOn/XOff output flow control flag}
  fCTS: Byte; {CTS hardware handshake flag}
  xOn: CHAR; {XOn character}
  xOff: CHAR; {XOff character}
  errs: Byte; {errors that cause abort}
  evts: Byte; {status changes that cause events}
  fInX: Byte; {XOn/XOff input flow control flag}
  null: Byte {not used}
END;

SerStaRec = PACKED RECORD
  cumErrs: Byte; {cumulative errors}
  xOffSent: Byte; {XOff sent as input flow control}
  rdPend: Byte; {read pending flag}
  wrPend: Byte; {write pending flag}
  ctsHold: Byte; {CTS flow control hold flag}
  xOffHold: Byte {XOff flow control hold flag}
END;

Routines [Not in ROM]

Opening and Closing the RAM Serial Driver

FUNCTION RAMSDOpen (whichPort: SPortSel) : OSErr;
PROCEDURE RAMSDClose (whichPort: SPortSel);

Changing Serial Driver Information

FUNCTION SerReset (refNum: INTEGER; serConfig: INTEGER) : OSErr;
FUNCTION SerSetBuf (refNum: INTEGER; serBPtr: Ptr; serBLen: INTEGER) : OSErr;
FUNCTION SerHShake (refNum: INTEGER; flags: SerShk) : OSErr;
FUNCTION SerSetBrk (refNum: INTEGER) : OSErr;
FUNCTION SerClrBrk (refNum: INTEGER) : OSErr;

Getting Serial Driver Information

FUNCTION SerGetBuf (refNum: INTEGER; VAR count: LONGINT) : OSErr;
FUNCTION SerStatus (refNum: INTEGER; VAR serSta: SerStaRec) : OSErr;
Advanced Control Calls (RAM Serial Driver)

<table>
<thead>
<tr>
<th>csCode</th>
<th>csParam</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>baudRate</td>
<td>Set baud rate (actual rate, as an integer)</td>
</tr>
<tr>
<td>19</td>
<td>char</td>
<td>Replace parity errors</td>
</tr>
<tr>
<td>21</td>
<td></td>
<td>Unconditionally set XOff for output flow control</td>
</tr>
<tr>
<td>22</td>
<td></td>
<td>Unconditionally clear XOff for input flow control</td>
</tr>
<tr>
<td>23</td>
<td></td>
<td>Send XOn for input flow control if XOff was sent last</td>
</tr>
<tr>
<td>24</td>
<td></td>
<td>Unconditionally send XOn for input flow control</td>
</tr>
<tr>
<td>25</td>
<td></td>
<td>Send XOff for input flow control if XOn was sent last</td>
</tr>
<tr>
<td>26</td>
<td></td>
<td>Unconditionally send XOff for input flow control</td>
</tr>
<tr>
<td>27</td>
<td></td>
<td>Reset SCC channel</td>
</tr>
</tbody>
</table>

Driver Names and Reference Numbers

<table>
<thead>
<tr>
<th>Driver</th>
<th>Driver name</th>
<th>Reference number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modem port input</td>
<td>.AIn</td>
<td>-6</td>
</tr>
<tr>
<td>Modem port output</td>
<td>.AOut</td>
<td>-7</td>
</tr>
<tr>
<td>Printer port input</td>
<td>.BIn</td>
<td>-8</td>
</tr>
<tr>
<td>Printer port output</td>
<td>.BOut</td>
<td>-9</td>
</tr>
</tbody>
</table>

Assembly-Language Information

Constants

; Result codes

noErr    .EQU 0  ;no error
openErr  .EQU -23 ;attempt to open RAM Serial Driver failed

Structure of Control Information for SerHShake

shFXOn   XOn/XOff output flow control flag (byte)
shFCTS   CTS hardware handshake flag (byte)
shXOn    XOn character (byte)
shXOff   XOff character (byte)
shErrs   Errors that cause abort (byte)
shEvts   Status changes that cause events (byte)
shFinX   XOn/XOff input flow control flag (byte)
Structure of Status Information for SerStatus

- **ssCumErrs**: Cumulative errors (byte)
- **ssXOffSent**: XOff sent as input flow control (byte)
- **ssRdPend**: Read pending flag (byte)
- **ssWrPend**: Write pending flag (byte)
- **ssCTSHold**: CTS flow control hold flag (byte)
- **ssXOffHold**: XOff flow control hold flag (byte)

**Equivalent Device Manager Calls**

<table>
<thead>
<tr>
<th>Pascal routine</th>
<th>Call</th>
</tr>
</thead>
<tbody>
<tr>
<td>SerReset</td>
<td>Control with csCode=8, csParam=serConfig</td>
</tr>
<tr>
<td>SerSetBuf</td>
<td>Control with csCode=8, csParam=serBPtr, csParam+4=serLen</td>
</tr>
<tr>
<td>SerHShake</td>
<td>Control with csCode=10, csParam through csParam+6=flags</td>
</tr>
<tr>
<td>SerSetBrk</td>
<td>Control with csCode=12</td>
</tr>
<tr>
<td>SerClrBrk</td>
<td>Control with csCode=11</td>
</tr>
<tr>
<td>SerGetBuf</td>
<td>Status with csCode=2; count returned in csParam</td>
</tr>
<tr>
<td>SerStatus</td>
<td>Status with csCode=8; serSta returned in csParam through csParam+5</td>
</tr>
</tbody>
</table>
SOUND DRIVER

Constants

CONST { Mode values for synthesizers }

- swMode = -1; {square-wave synthesizer}
- ftMode = 1; {four-tone synthesizer}
- ffMode = 0; {free-form synthesizer}

Data Types

TYPE { Free-form synthesizer }

FFSynthPtr = "FFSynthRec;
FFSynthRec = RECORD
  mode: INTEGER; {always ffMode}
  count: Fixed;{"sampling" factor}
  waveBytes: FreeWave {waveform description}
END;

FreeWave = PACKED ARRAY[0..30000] OF Byte;

{ Square-wave synthesizer }

SWSynthPtr = "SWSynthRec;
SWSynthRec = RECORD
  mode: INTEGER; {always swMode}
  triplets: Tones {sounds}
END;

Tones = ARRAY[0..5000] OF Tone;
Tone = RECORD
  count: INTEGER; {frequency}
  amplitude: INTEGER; {amplitude, 0-255}
  duration: INTEGER {duration in ticks}
END;

{ Four-tone synthesizer }

FTSynthPtr = "FTSynthRec;
FTSynthRec = RECORD
  mode: INTEGER; {always ftMode}
  sndRec: FTSndRecPtr {tones to play}
END;
Inside Macintosh

FTSndRecPtr = ^FTSoundRec;
FTSoundRec = RECORD
  duration: INTEGER; {duration in ticks}
sound1Rate: Fixed; {tone 1 cycle rate}
sound1Phase: LONGINT; {tone 1 byte offset}
sound2Rate: Fixed; {tone 2 cycle rate}
sound2Phase: LONGINT; {tone 2 byte offset}
sound3Rate: Fixed; {tone 3 cycle rate}
sound3Phase: LONGINT; {tone 3 byte offset}
sound4Rate: Fixed; {tone 4 cycle rate}
sound4Phase: LONGINT; {tone 4 byte offset}
sound1Wave: WavePtr; {tone 1 waveform}
sound2Wave: WavePtr; {tone 2 waveform}
sound3Wave: WavePtr; {tone 3 waveform}
sound4Wave: WavePtr {tone 4 waveform}
END;

WavePtr = ^Wave;
Wave = PACKED ARRAY[0..255] OF Byte;

Routines [Not in ROM]

PROCEDURE StartSound (synthRec: Ptr; numBytes: LONGINT; completionRtn: ProcPtr);
PROCEDURE StopSound;
FUNCTION SoundDone : BOOLEAN;
PROCEDURE GetSoundVol (VAR level: INTEGER);
PROCEDURE SetSoundVol (level: INTEGER);

Assembly-Language Information

Routines

Pascal name        Equivalent for assembly language
StartSound         Call Write with ioRefNum=-4, ioBuffer=synthRec, ioReqCount=numBytes
StopSound          Call KillIO and (for square-wave) set CurPitch to 0
SoundDone          Poll ioResult field of most recent Write call's parameter block
GetSoundVol        Get low-order three bits of variable SdVolume
SetSoundVol        Call this Pascal procedure from your program

Variables

SdVolume           Speaker volume (byte: low-order three bits only)
SoundPtr           Pointer to four-tone record
SoundLevel         Amplitude in 740-byte buffer (byte)
CurPitch           Value of count in square-wave synthesizer buffer (word)

III-166 Sound Driver
Sound Driver Values for Notes

The following table contains values for the rate field of a four-tone synthesizer and the count field of a square-wave synthesizer. A just-tempered scale—in the key of C, as an example—is given in the first four columns; you can use a just-tempered scale for perfect tuning in a particular key. The last four columns give an equal-tempered scale, for applications that may use any key; this scale is appropriate for most Macintosh sound applications. Following this table is a list of the ratios used in calculating these values, and instructions on how to calculate them for a just-tempered scale in any key.

<table>
<thead>
<tr>
<th>Note</th>
<th>Rate for Four-Tone</th>
<th>Count for Square-Wave</th>
<th>Rate for Four-Tone</th>
<th>Count for Square-Wave</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Long</td>
<td>Fixed</td>
<td>Word</td>
<td>Integer</td>
</tr>
<tr>
<td>3 octaves below middle C</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>612B</td>
<td>0.37956</td>
<td>5CBA</td>
<td>23738</td>
</tr>
<tr>
<td>C#</td>
<td>667C</td>
<td>0.40033</td>
<td>57EB</td>
<td>22507</td>
</tr>
<tr>
<td>Db</td>
<td>67A6</td>
<td>0.40488</td>
<td>56EF</td>
<td>22255</td>
</tr>
<tr>
<td>D</td>
<td>6D51</td>
<td>0.42702</td>
<td>526D</td>
<td>21101</td>
</tr>
<tr>
<td>Ebb</td>
<td>6E8F</td>
<td>0.43187</td>
<td>5180</td>
<td>20864</td>
</tr>
<tr>
<td>D#</td>
<td>71DF</td>
<td>0.44481</td>
<td>4F21</td>
<td>20257</td>
</tr>
<tr>
<td>Eb</td>
<td>749A</td>
<td>0.45547</td>
<td>4D46</td>
<td>19782</td>
</tr>
<tr>
<td>E</td>
<td>7976</td>
<td>0.47446</td>
<td>4A2F</td>
<td>18991</td>
</tr>
<tr>
<td>F</td>
<td>818F</td>
<td>0.50609</td>
<td>458C</td>
<td>17804</td>
</tr>
<tr>
<td>F#</td>
<td>88A5</td>
<td>0.53377</td>
<td>41F0</td>
<td>16880</td>
</tr>
<tr>
<td>Gb</td>
<td>8A32</td>
<td>0.53983</td>
<td>4133</td>
<td>16691</td>
</tr>
<tr>
<td>G</td>
<td>91C1</td>
<td>0.56935</td>
<td>3DD1</td>
<td>15825</td>
</tr>
<tr>
<td>G#</td>
<td>97D4</td>
<td>0.59308</td>
<td>3B58</td>
<td>15192</td>
</tr>
<tr>
<td>Ab</td>
<td>9B79</td>
<td>0.60732</td>
<td>39F4</td>
<td>14836</td>
</tr>
<tr>
<td>A</td>
<td>A1F3</td>
<td>0.63261</td>
<td>37A3</td>
<td>14243</td>
</tr>
<tr>
<td>Bbb</td>
<td>A3CA</td>
<td>0.63980</td>
<td>3703</td>
<td>14083</td>
</tr>
<tr>
<td>A#</td>
<td>AA0C</td>
<td>0.66425</td>
<td>34FD</td>
<td>13565</td>
</tr>
<tr>
<td>Bb</td>
<td>ACBF</td>
<td>0.67479</td>
<td>3429</td>
<td>13353</td>
</tr>
<tr>
<td>B</td>
<td>B631</td>
<td>0.71169</td>
<td>3174</td>
<td>12660</td>
</tr>
</tbody>
</table>

2 octaves below middle C

<table>
<thead>
<tr>
<th>Note</th>
<th>Rate for Four-Tone</th>
<th>Count for Square-Wave</th>
<th>Rate for Four-Tone</th>
<th>Count for Square-Wave</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>C257</td>
<td>0.75914</td>
<td>2E5D</td>
<td>11869</td>
</tr>
<tr>
<td>C#</td>
<td>CCF8</td>
<td>0.80066</td>
<td>2BF6</td>
<td>11254</td>
</tr>
<tr>
<td>Db</td>
<td>CF4C</td>
<td>0.80975</td>
<td>2B77</td>
<td>11127</td>
</tr>
<tr>
<td>D</td>
<td>DAA2</td>
<td>0.85403</td>
<td>2936</td>
<td>10550</td>
</tr>
<tr>
<td>Ebb</td>
<td>DD1D</td>
<td>0.86372</td>
<td>28C0</td>
<td>10432</td>
</tr>
<tr>
<td>D#</td>
<td>E3BE</td>
<td>0.88962</td>
<td>2790</td>
<td>10128</td>
</tr>
<tr>
<td>Eb</td>
<td>E935</td>
<td>0.91096</td>
<td>26A3</td>
<td>9891</td>
</tr>
<tr>
<td>E</td>
<td>F2ED</td>
<td>0.94893</td>
<td>2517</td>
<td>9495</td>
</tr>
<tr>
<td>F</td>
<td>1031E</td>
<td>1.01218</td>
<td>22C6</td>
<td>8902</td>
</tr>
<tr>
<td>F#</td>
<td>1114A</td>
<td>1.06754</td>
<td>20F8</td>
<td>8440</td>
</tr>
<tr>
<td>Gb</td>
<td>11465</td>
<td>1.07967</td>
<td>2099</td>
<td>8345</td>
</tr>
<tr>
<td>G</td>
<td>12382</td>
<td>1.13870</td>
<td>1EE9</td>
<td>7913</td>
</tr>
</tbody>
</table>

Sound Driver III-167
### Inside Macintosh

<table>
<thead>
<tr>
<th>Note</th>
<th>Long Fixed Word</th>
<th>Integer</th>
<th>Long Fixed Word Integer</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>2 octaves below middle C</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G#</td>
<td>12FA8 1.18616 1DAC</td>
<td>7596</td>
<td>131B8 1.19421 1D79</td>
</tr>
<tr>
<td>Ab</td>
<td>136F1 1.21461 1CFA</td>
<td>7418</td>
<td>143E6 1.26523 1BD1</td>
</tr>
<tr>
<td>A</td>
<td>143E6 1.26523 1BD1</td>
<td>7121</td>
<td>143E6 1.26523 1BD1</td>
</tr>
<tr>
<td>Bbb</td>
<td>14794 1.27960 1B81</td>
<td>7041</td>
<td>15729 1.34047 1A42</td>
</tr>
<tr>
<td>A#</td>
<td>15418 1.32849 1A7E</td>
<td>6782</td>
<td>143E6 1.26523 1BD1</td>
</tr>
<tr>
<td>Bb</td>
<td>1597E 1.34958 1A14</td>
<td>6676</td>
<td>143E6 1.26523 1BD1</td>
</tr>
<tr>
<td>B</td>
<td>16C63 1.42339 18BA</td>
<td>6330</td>
<td>16B90 1.42017 18C8</td>
</tr>
<tr>
<td><strong>1 octave below middle C</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>184AE 1.51828 172F</td>
<td>5935</td>
<td>1812F 1.50462 1764</td>
</tr>
<tr>
<td>C#</td>
<td>199EF 1.60130 15FB</td>
<td>5627</td>
<td>19816 1.59409 1614</td>
</tr>
<tr>
<td>Db</td>
<td>19E97 1.61949 15BC</td>
<td>5564</td>
<td>1B05A 1.68887 14D7</td>
</tr>
<tr>
<td>D</td>
<td>1B543 1.70805 149B</td>
<td>5275</td>
<td>1CFA 7418</td>
</tr>
<tr>
<td>Ebb</td>
<td>1BA33 1.72746 1460</td>
<td>5216</td>
<td>1B05A 1.68887 14D7</td>
</tr>
<tr>
<td>D#</td>
<td>1C77B 1.77922 13C8</td>
<td>5064</td>
<td>1CA10 1.78931 13AC</td>
</tr>
<tr>
<td>Eb</td>
<td>1D26A 1.82193 1351</td>
<td>4945</td>
<td>1E54D 1.89571 1291</td>
</tr>
<tr>
<td>E</td>
<td>1E5D9 1.89784 128C</td>
<td>4748</td>
<td>1E54D 1.89571 1291</td>
</tr>
<tr>
<td>F</td>
<td>2063D 2.02437 1163</td>
<td>4451</td>
<td>20228 2.00842 1186</td>
</tr>
<tr>
<td>F#</td>
<td>2229F 2.13507 107C</td>
<td>4220</td>
<td>220BB 2.12785 108A</td>
</tr>
<tr>
<td>Gb</td>
<td>228CA 2.15932 104D</td>
<td>4173</td>
<td>220BB 2.12785 108A</td>
</tr>
<tr>
<td>G</td>
<td>2470F 2.27740 1F7</td>
<td>3956</td>
<td>2411F 2.25438 F9D</td>
</tr>
<tr>
<td>G#</td>
<td>25F4F 2.37230 1D6</td>
<td>3798</td>
<td>26370 2.38843 EBC</td>
</tr>
<tr>
<td>Ab</td>
<td>26DE3 2.42924 1E7D</td>
<td>3709</td>
<td>26370 2.38843 EBC</td>
</tr>
<tr>
<td>A</td>
<td>287CC 2.53046 1D9</td>
<td>3561</td>
<td>287CC 2.53046 DE9</td>
</tr>
<tr>
<td>Bbb</td>
<td>28F28 2.55920 1DC1</td>
<td>3521</td>
<td>287CC 2.53046 DE9</td>
</tr>
<tr>
<td>A#</td>
<td>2A830 2.65698 1D3F</td>
<td>3391</td>
<td>2AE51 2.68092 D21</td>
</tr>
<tr>
<td>Bb</td>
<td>2B2FC 2.69916 1D0A</td>
<td>3338</td>
<td>2AE51 2.68092 D21</td>
</tr>
<tr>
<td>B</td>
<td>2D8C6 2.84677 1C5D</td>
<td>3165</td>
<td>2D721 2.84035 C64</td>
</tr>
</tbody>
</table>

### Middle C

<table>
<thead>
<tr>
<th>Note</th>
<th>Long Fixed Word</th>
<th>Integer</th>
<th>Long Fixed Word Integer</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>3095B 3.03654 1B9</td>
<td>2967</td>
<td>3025D 3.00923 BB2</td>
</tr>
<tr>
<td>C#</td>
<td>333DE 3.20261 AFD</td>
<td>2813</td>
<td>3302C 3.18817 B0A</td>
</tr>
<tr>
<td>Db</td>
<td>332D2 3.23898 ADE</td>
<td>2782</td>
<td>360B5 3.37776 A6C</td>
</tr>
<tr>
<td>D</td>
<td>36A87 3.41612 A4E</td>
<td>2638</td>
<td>360B5 3.37776 A6C</td>
</tr>
<tr>
<td>Ebb</td>
<td>37476 3.45493 A30</td>
<td>2608</td>
<td>360B5 3.37776 A6C</td>
</tr>
<tr>
<td>D#</td>
<td>38EF7 3.55846 9E4</td>
<td>2532</td>
<td>39420 3.57861 9D6</td>
</tr>
<tr>
<td>Eb</td>
<td>3A4D4 3.64386 9A9</td>
<td>2473</td>
<td>3CA99 3.79140 949</td>
</tr>
<tr>
<td>E</td>
<td>3CBB2 3.79568 946</td>
<td>2374</td>
<td>3CA99 3.79140 949</td>
</tr>
<tr>
<td>F</td>
<td>40C7A 4.04874 8B1</td>
<td>2225</td>
<td>40450 4.01685 8C3</td>
</tr>
<tr>
<td>F#</td>
<td>44328 4.27014 83E</td>
<td>2110</td>
<td>44176 4.25571 845</td>
</tr>
<tr>
<td>Gb</td>
<td>45193 4.31865 826</td>
<td>2086</td>
<td>44176 4.25571 845</td>
</tr>
<tr>
<td>G#</td>
<td>4BE9F 4.74461 76B</td>
<td>1899</td>
<td>4CE61 4.77687 75E</td>
</tr>
<tr>
<td>Ab</td>
<td>4DBC5 4.85847 73F</td>
<td>1855</td>
<td>4CE61 4.77687 75E</td>
</tr>
<tr>
<td>A</td>
<td>50F98 5.06091 6F4</td>
<td>1780</td>
<td>50F98 5.06091 6F4</td>
</tr>
</tbody>
</table>

### III-168 Sound Driver
<table>
<thead>
<tr>
<th>Note</th>
<th>Long</th>
<th>Fixed</th>
<th>Word</th>
<th>Integer</th>
<th>Long</th>
<th>Fixed</th>
<th>Word</th>
<th>Integer</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Middle C</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bbb</td>
<td>51E4F</td>
<td>5.11839</td>
<td>6E0</td>
<td>1760</td>
<td>55CA2</td>
<td>5.36185</td>
<td>690</td>
<td>1680</td>
</tr>
<tr>
<td>A#</td>
<td>55060</td>
<td>5.31396</td>
<td>6A0</td>
<td>1696</td>
<td>66059</td>
<td>6.37636</td>
<td>585</td>
<td>1413</td>
</tr>
<tr>
<td>Bb</td>
<td>565F8</td>
<td>5.39383</td>
<td>685</td>
<td>1669</td>
<td>55CA2</td>
<td>5.36185</td>
<td>690</td>
<td>1680</td>
</tr>
<tr>
<td>B</td>
<td>5B18B</td>
<td>5.69353</td>
<td>62F</td>
<td>1583</td>
<td>5AE41</td>
<td>5.68068</td>
<td>632</td>
<td>1586</td>
</tr>
<tr>
<td><strong>1 octave above middle C</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>612B7</td>
<td>6.07310</td>
<td>5CC</td>
<td>1484</td>
<td>604BB</td>
<td>6.01848</td>
<td>5D9</td>
<td>1497</td>
</tr>
<tr>
<td>C#</td>
<td>667BD</td>
<td>6.40523</td>
<td>57F</td>
<td>1407</td>
<td>66059</td>
<td>6.37636</td>
<td>585</td>
<td>1413</td>
</tr>
<tr>
<td>Db</td>
<td>67A5C</td>
<td>6.47797</td>
<td>56F</td>
<td>1391</td>
<td>66059</td>
<td>6.37636</td>
<td>585</td>
<td>1413</td>
</tr>
<tr>
<td>D</td>
<td>6D50D</td>
<td>6.83223</td>
<td>527</td>
<td>1319</td>
<td>6C169</td>
<td>6.75551</td>
<td>536</td>
<td>1334</td>
</tr>
<tr>
<td>Ebb</td>
<td>6E8EB</td>
<td>6.90984</td>
<td>518</td>
<td>1304</td>
<td>6C169</td>
<td>6.75551</td>
<td>536</td>
<td>1334</td>
</tr>
<tr>
<td>D#</td>
<td>71DEE</td>
<td>7.11691</td>
<td>4F2</td>
<td>1266</td>
<td>7283F</td>
<td>7.15721</td>
<td>4EB</td>
<td>1259</td>
</tr>
<tr>
<td>Eb</td>
<td>749A8</td>
<td>7.28772</td>
<td>4D4</td>
<td>1236</td>
<td>749A8</td>
<td>7.28772</td>
<td>4D4</td>
<td>1236</td>
</tr>
<tr>
<td>E</td>
<td>79764</td>
<td>7.59137</td>
<td>4A3</td>
<td>1187</td>
<td>79533</td>
<td>7.58281</td>
<td>4A4</td>
<td>1188</td>
</tr>
<tr>
<td>F</td>
<td>818F3</td>
<td>8.09746</td>
<td>459</td>
<td>1113</td>
<td>808A1</td>
<td>8.03371</td>
<td>462</td>
<td>112.2</td>
</tr>
<tr>
<td>F#</td>
<td>88A51</td>
<td>8.54030</td>
<td>41F</td>
<td>1055</td>
<td>882EC</td>
<td>8.51141</td>
<td>423</td>
<td>1059</td>
</tr>
<tr>
<td>Gb</td>
<td>8A326</td>
<td>8.63730</td>
<td>413</td>
<td>1043</td>
<td>8A326</td>
<td>8.63730</td>
<td>413</td>
<td>1043</td>
</tr>
<tr>
<td>G</td>
<td>91C12</td>
<td>9.10965</td>
<td>3DD</td>
<td>989</td>
<td>9047D</td>
<td>9.01753</td>
<td>3E7</td>
<td>999</td>
</tr>
<tr>
<td>G#</td>
<td>97D3D</td>
<td>9.48921</td>
<td>3B6</td>
<td>950</td>
<td>98DC2</td>
<td>9.55374</td>
<td>3AF</td>
<td>943</td>
</tr>
<tr>
<td>Ab</td>
<td>9B78B</td>
<td>9.71696</td>
<td>39F</td>
<td>927</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>A1F30</td>
<td>10.12183</td>
<td>37A</td>
<td>890</td>
<td>A1F30</td>
<td>10.12183</td>
<td>37A</td>
<td>890</td>
</tr>
<tr>
<td>Bbb</td>
<td>A3C9F</td>
<td>10.23680</td>
<td>370</td>
<td>880</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A#</td>
<td>AA0BF</td>
<td>10.62791</td>
<td>350</td>
<td>848</td>
<td>AB945</td>
<td>10.72371</td>
<td>348</td>
<td>840</td>
</tr>
<tr>
<td>Bb</td>
<td>ACBEF</td>
<td>10.79662</td>
<td>343</td>
<td>835</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>B6316</td>
<td>11.38705</td>
<td>317</td>
<td>791</td>
<td>B5C83</td>
<td>11.36137</td>
<td>319</td>
<td>793</td>
</tr>
<tr>
<td><strong>2 octaves above middle C</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>C256D</td>
<td>12.14619</td>
<td>2E6</td>
<td>742</td>
<td>C0976</td>
<td>12.03696</td>
<td>2ED</td>
<td>749</td>
</tr>
<tr>
<td>C#</td>
<td>CCF79</td>
<td>12.81044</td>
<td>2BF</td>
<td>703</td>
<td>CC0B1</td>
<td>12.75270</td>
<td>2C3</td>
<td>707</td>
</tr>
<tr>
<td>Db</td>
<td>CF4B9</td>
<td>12.95595</td>
<td>2B7</td>
<td>695</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>DAA1B</td>
<td>13.66447</td>
<td>293</td>
<td>659</td>
<td>D82D2</td>
<td>13.51102</td>
<td>29B</td>
<td>667</td>
</tr>
<tr>
<td>Ebb</td>
<td>DD1D6</td>
<td>13.81967</td>
<td>28C</td>
<td>652</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D#</td>
<td>E3BDC</td>
<td>14.23833</td>
<td>279</td>
<td>633</td>
<td>E507E</td>
<td>14.31442</td>
<td>275</td>
<td>629</td>
</tr>
<tr>
<td>Eb</td>
<td>E9350</td>
<td>14.57544</td>
<td>26A</td>
<td>618</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>F2EC8</td>
<td>15.18274</td>
<td>251</td>
<td>593</td>
<td>P2A65</td>
<td>15.16560</td>
<td>252</td>
<td>594</td>
</tr>
<tr>
<td>F</td>
<td>1031E7</td>
<td>16.19493</td>
<td>22C</td>
<td>556</td>
<td>1011A1</td>
<td>16.06740</td>
<td>231</td>
<td>561</td>
</tr>
<tr>
<td>F#</td>
<td>1114A1</td>
<td>17.08058</td>
<td>210</td>
<td>528</td>
<td>1105D8</td>
<td>17.02283</td>
<td>211</td>
<td>529</td>
</tr>
<tr>
<td>Gb</td>
<td>11464C</td>
<td>17.27460</td>
<td>20A</td>
<td>522</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>12382A</td>
<td>18.21930</td>
<td>1EF</td>
<td>495</td>
<td>1208F9</td>
<td>18.03505</td>
<td>1F4</td>
<td>500</td>
</tr>
<tr>
<td>G#</td>
<td>12FA7B</td>
<td>18.97844</td>
<td>1DB</td>
<td>475</td>
<td>131B83</td>
<td>19.10747</td>
<td>1D8</td>
<td>472</td>
</tr>
<tr>
<td>Ab</td>
<td>136F15</td>
<td>19.43391</td>
<td>1D0</td>
<td>464</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>143E61</td>
<td>20.24367</td>
<td>1BD</td>
<td>445</td>
<td>143E61</td>
<td>20.24367</td>
<td>1BD</td>
<td>445</td>
</tr>
<tr>
<td>Bbb</td>
<td>14793D</td>
<td>20.47359</td>
<td>1B8</td>
<td>440</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A#</td>
<td>15417F</td>
<td>21.25584</td>
<td>1A8</td>
<td>424</td>
<td>15728A</td>
<td>21.44742</td>
<td>1A4</td>
<td>420</td>
</tr>
<tr>
<td>Bb</td>
<td>1597DE</td>
<td>21.59323</td>
<td>1A1</td>
<td>417</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>16C62D</td>
<td>22.77412</td>
<td>18C</td>
<td>396</td>
<td>16B906</td>
<td>22.72275</td>
<td>18D</td>
<td>397</td>
</tr>
</tbody>
</table>

*Sound Driver III-169*
The following table gives the ratios used in calculating the above values. It shows the relationship between the notes making up the just-tempered scale in the key of C; should you need to implement a just-tempered scale in some other key, you can do so as follows: First get the value of the root note in the proper octave in the equal-tempered scale (from the above table). Then use the following table to determine the values of the intervals for the other notes in the key by multiplying the ratio by the root note.

<table>
<thead>
<tr>
<th>Chromatic interval</th>
<th>Note</th>
<th>Just-tempered frequency ratio</th>
<th>Equal-tempered frequency ratio</th>
<th>Interval type</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>C</td>
<td>1.00000</td>
<td>1.00000</td>
<td>Unison</td>
</tr>
<tr>
<td>1</td>
<td>C#</td>
<td>1.05469</td>
<td>1.05946</td>
<td>Minor second as chromatic semitone</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Minor second as diatonic semitone</td>
</tr>
<tr>
<td>2</td>
<td>D</td>
<td>1.11111</td>
<td>1.12246</td>
<td>Major second as minor tone</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Major second as major tone</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Diminished third</td>
</tr>
<tr>
<td>3</td>
<td>D#</td>
<td>1.17188</td>
<td>1.18921</td>
<td>Augmented second</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Minor third</td>
</tr>
<tr>
<td>4</td>
<td>E</td>
<td>1.25000</td>
<td>1.25992</td>
<td>Major third</td>
</tr>
<tr>
<td>5</td>
<td>F</td>
<td>1.33333</td>
<td>1.33484</td>
<td>Fourth</td>
</tr>
<tr>
<td>6</td>
<td>F#</td>
<td>1.40625</td>
<td>1.41421</td>
<td>Tritone as augmented fourth</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Tritone as diminished fifth</td>
</tr>
<tr>
<td>7</td>
<td>G</td>
<td>1.50000</td>
<td>1.49831</td>
<td>Fifth</td>
</tr>
<tr>
<td>Chromatic interval</td>
<td>Note</td>
<td>Just-tempered frequency ratio</td>
<td>Equal-tempered frequency ratio</td>
<td>Interval type</td>
</tr>
<tr>
<td>-------------------</td>
<td>------</td>
<td>------------------------------</td>
<td>-------------------------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>8</td>
<td>G#</td>
<td>1.56250</td>
<td>1.58740</td>
<td>Augmented fifth</td>
</tr>
<tr>
<td></td>
<td>Ab</td>
<td>1.60000</td>
<td></td>
<td>Minor sixth</td>
</tr>
<tr>
<td>9</td>
<td>A</td>
<td>1.66667</td>
<td>1.68179</td>
<td>Major sixth</td>
</tr>
<tr>
<td></td>
<td>Bbb</td>
<td>1.68560</td>
<td></td>
<td>Diminished seventh</td>
</tr>
<tr>
<td>10</td>
<td>A#</td>
<td>1.75000</td>
<td>1.78180</td>
<td>Augmented sixth</td>
</tr>
<tr>
<td></td>
<td>Bb</td>
<td>1.77778</td>
<td></td>
<td>Minor seventh</td>
</tr>
<tr>
<td>11</td>
<td>B</td>
<td>1.87500</td>
<td>1.88775</td>
<td>Major seventh</td>
</tr>
<tr>
<td>12</td>
<td>C</td>
<td>2.00000</td>
<td>2.00000</td>
<td>Octave</td>
</tr>
</tbody>
</table>

*Sound Driver III-171*
STANDARD FILE PACKAGE

Constants

CONST { SFPutFile dialog template ID }

putDlgID = -3999;

{ Item numbers of enabled items in SFPutFile dialog }

putSave = 1; {Save button}
putCancel = 2; {Cancel button}
putEject = 5; {Eject button}
putDrive = 6; {Drive button}
putName = 7; {editText item for file name}

{ SFGetFile dialog template ID }

getDlgID = -4000;

{ Item numbers of enabled items in SFGetFile dialog }

getOpen = 1; {Open button}
getCancel = 3; {Cancel button}
getEject = 5; {Eject button}
getDrive = 6; {Drive button}
getNmList = 7; {userItem for file name list}
getScroll = 8; {userItem for scroll bar}

Data Types

TYPE SFReply = RECORD
  good: BOOLEAN; {FALSE if ignore command}
copy: BOOLEAN; {not used}
ftype: OSTYPE; {file type or not used}
vRefNum: INTEGER; {volume reference number}
version: INTEGER; {file's version number}
fname: STRING[63] {file name}
END;

SFTypeList = ARRAY[0..3] OF OSTYPE;
Routines

PROCEDURE SFPutFile (where: Point; prompt: Str255; origName: Str255;
dlgHook: ProcPtr; VAR reply: SFReply);

PROCEDURE SFPPutFile (where: Point; prompt: Str255; origName: Str255;
dlgHook: ProcPtr; VAR reply: SFReply; dlgID: INTEGER; filterProc: ProcPtr);

PROCEDURE SFGetFile (where: Point; prompt: Str255; fileFilter: ProcPtr;
numTypes: INTEGER; typeList: SFTypeList; dlgHook: ProcPtr; VAR reply: SFReply);

PROCEDURE SFPGetFile (where: Point; prompt: Str255; fileFilter: ProcPtr;
numTypes: INTEGER; typeList: SFTypeList; dlgHook: ProcPtr; VAR reply: SFReply; dlgID: INTEGER;
filterProc: ProcPtr);

DlgHook Function

FUNCTION MyDlg (item: INTEGER; theDialog: DialogPtr) : INTEGER;

FileFilter Function

FUNCTION MyFileFilter (paramBlock: ParmBlkPtr) : BOOLEAN;

Standard SFPutFile Items

<table>
<thead>
<tr>
<th>Item number</th>
<th>Item</th>
<th>Standard display rectangle</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Save button</td>
<td>(12,74)(82,92)</td>
</tr>
<tr>
<td>2</td>
<td>Cancel button</td>
<td>(114,74)(184,92)</td>
</tr>
<tr>
<td>3</td>
<td>Prompt string (statText)</td>
<td>(12,12)(184,28)</td>
</tr>
<tr>
<td>4</td>
<td>UserItem for disk name</td>
<td>(209,16)(295,34)</td>
</tr>
<tr>
<td>5</td>
<td>Eject button</td>
<td>(217,43)(287,61)</td>
</tr>
<tr>
<td>6</td>
<td>Drive button</td>
<td>(217,74)(287,92)</td>
</tr>
<tr>
<td>7</td>
<td>EditText item for file name</td>
<td>(14,34)(182,50)</td>
</tr>
<tr>
<td>8</td>
<td>UserItem for dotted line</td>
<td>(200,16)(201,88)</td>
</tr>
</tbody>
</table>
Resource IDs of SFPutFile Alerts

<table>
<thead>
<tr>
<th>Alert</th>
<th>Resource ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disk not found</td>
<td>-3994</td>
</tr>
<tr>
<td>System error</td>
<td>-3995</td>
</tr>
<tr>
<td>Existing file</td>
<td>-3996</td>
</tr>
<tr>
<td>Locked disk</td>
<td>-3997</td>
</tr>
</tbody>
</table>

Standard SFGetFile Items

<table>
<thead>
<tr>
<th>Item number</th>
<th>Item</th>
<th>Standard display rectangle</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Open button</td>
<td>(152,28)(232,46)</td>
</tr>
<tr>
<td>2</td>
<td>Invisible button</td>
<td>(1152,28)(1232,77)</td>
</tr>
<tr>
<td>3</td>
<td>Cancel button</td>
<td>(152,90)(232,108)</td>
</tr>
<tr>
<td>4</td>
<td>UserItem for disk name</td>
<td>(248,28)(344,46)</td>
</tr>
<tr>
<td>5</td>
<td>Eject button</td>
<td>(256,59)(336,77)</td>
</tr>
<tr>
<td>6</td>
<td>Drive button</td>
<td>(256,90)(336,108)</td>
</tr>
<tr>
<td>7</td>
<td>UserItem for file name list</td>
<td>(12,11)(125,125)</td>
</tr>
<tr>
<td>8</td>
<td>UserItem for scroll bar</td>
<td>(124,11)(140,125)</td>
</tr>
<tr>
<td>9</td>
<td>UserItem for dotted line</td>
<td>(244,20)(245,116)</td>
</tr>
<tr>
<td>10</td>
<td>Invisible text (statText)</td>
<td>(1044,20)(1145,116)</td>
</tr>
</tbody>
</table>

Assembly-Language Information

Constants

; SFPutFile dialog template ID
putDlgID .EQU -3999

; Item numbers of enabled items in SFPutFile dialog
putSave  .EQU 1 ; Save button
putCancel .EQU 2 ; Cancel button
putEject  .EQU 5 ; Eject button
putDrive  .EQU 6 ; Drive button
putName   .EQU 7 ; editText item for file name

; SFGetFile dialog template ID
getDlgID   .EQU -4000
; Item numbers of enabled items in SFGetFile dialog

getOpen .EQU 1 ;Open button
getCancel .EQU 3 ;Cancel button
getEject .EQU 5 ;Eject button
getDrive .EQU 6 ;Drive button
getNmList .EQU 7 ;userItem for file name list
getScroll .EQU 8 ;userItem for scroll bar

; Routine selectors
sfPutFile .EQU 1
sfGetFile .EQU 2
sfPPutFile .EQU 3
sfPGetFile .EQU 4

Reply Record Data Structure

rGood 0 if ignore command (byte)
rType File type (long)
rVolume Volume reference number (word)
rVersion File's version number (word)
rName File name (length byte followed by up to 63 characters)

Trap Macro Name

Pack3

Variables

SFSaveDisk Negative of volume reference number used by Standard File Package (word)
### SYSTEM ERROR HANDLER

#### Routines

PROCEDURE SysError (errorCode: INTEGER);

#### User Alerts

<table>
<thead>
<tr>
<th>ID</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Bus error: Invalid memory reference; happens only on a Macintosh XL</td>
</tr>
<tr>
<td>2</td>
<td>Address error: Word or long-word reference made to an odd address</td>
</tr>
<tr>
<td>3</td>
<td>Illegal instruction: The MC68000 received an instruction it didn't recognize.</td>
</tr>
<tr>
<td>4</td>
<td>Zero divide: Signed Divide (DIVS) or Unsigned Divide (DIVU) instruction with a divisor of 0 was executed.</td>
</tr>
<tr>
<td>5</td>
<td>Check exception: Check Register Against Bounds (CHK) instruction was executed and failed. Pascal &quot;value out of range&quot; errors are usually reported in this way.</td>
</tr>
<tr>
<td>6</td>
<td>TrapV exception: Trap On Overflow (TRAPV) instruction was executed and failed.</td>
</tr>
<tr>
<td>7</td>
<td>Privilege violation: Macintosh always runs in supervisor mode; perhaps an erroneous Return From Execution (RTE) instruction was executed.</td>
</tr>
<tr>
<td>8</td>
<td>Trace exception: The trace bit in the status register is set.</td>
</tr>
<tr>
<td>9</td>
<td>Line 1010 exception: The 1010 trap dispatcher has failed.</td>
</tr>
<tr>
<td>10</td>
<td>Line 1111 exception: Unimplemented instruction</td>
</tr>
<tr>
<td>11</td>
<td>Miscellaneous exception: All other MC68000 exceptions</td>
</tr>
<tr>
<td>12</td>
<td>Unimplemented core routine: An unimplemented trap number was encountered.</td>
</tr>
<tr>
<td>13</td>
<td>Spurious interrupt: The interrupt vector table entry for a particular level of interrupt is NIL; usually occurs with level 4, 5, 6, or 7 interrupts.</td>
</tr>
<tr>
<td>14</td>
<td>I/O system error: The File Manager is attempting to dequeue an entry from an I/O request queue that has a bad queue type field; perhaps the queue entry is unlocked. Or, the dCtlQHead field was NIL during a Fetch or Stash call. Or, a needed device control entry has been purged.</td>
</tr>
<tr>
<td>15</td>
<td>Segment Loader error: A GetResource call to read a segment into memory failed.</td>
</tr>
<tr>
<td>16</td>
<td>Floating point error: The halt bit in the floating-point environment word was set.</td>
</tr>
<tr>
<td>17-24</td>
<td>Can't load package: A GetResource call to read a package into memory failed.</td>
</tr>
<tr>
<td>25</td>
<td>Can't allocate requested memory block in the heap</td>
</tr>
<tr>
<td>26</td>
<td>Segment Loader error: A GetResource call to read 'CODE' resource 0 into memory failed; usually indicates a nonexecutable file.</td>
</tr>
</tbody>
</table>
Summary

27 File map destroyed: A logical block number was found that's greater than the number of the last logical block on the volume or less than the logical block number of the first allocation block on the volume.

28 Stack overflow error: The stack has expanded into the heap.

30 "Please insert the disk:" File Manager alert

41 The file named "Finder" can't be found on the disk.

100 Can't mount system startup volume. The system couldn't read the system resource file into memory.

32767 "Sorry, a system error occurred": Default alert message

System Startup Alerts

"Welcome to Macintosh"
"Disassembler installed"
"MacsBug installed"
"Warning—this startup disk is not usable"

Assembly-Language Information

Constants

; System error IDs

dsBusError .EQU 1 ; bus error
dsAddressErr .EQU 2 ; address error
dsIllInstErr .EQU 3 ; illegal instruction
dsZeroDivErr .EQU 4 ; zero divide
dsChkErr .EQU 5 ; check exception
dsOvflowErr .EQU 6 ; trapV exception
dsPrivErr .EQU 7 ; privilege violation
dsTraceErr .EQU 8 ; trace exception
dsLineAErr .EQU 9 ; line 1010 exception
dsLineFErr .EQU 10 ; line 1111 exception
dsMiscErr .EQU 11 ; miscellaneous exception
dsCoreErr .EQU 12 ; unimplemented core routine
dsIrqErr .EQU 13 ; spurious interrupt
dsIOCoreErr .EQU 14 ; I/O system error
dsLoadErr .EQU 15 ; Segment Loader error
dsFPErr .EQU 16 ; floating point error
dsNoPackErr .EQU 17 ; can't load package 0
dsNoPk1 .EQU 18 ; can't load package 1
dsNoPk2 .EQU 19 ; can't load package 2
dsNoPk3 .EQU 20 ; can't load package 3
dsNoPk4 .EQU 21 ; can't load package 4
dsNoPk5 .EQU 22 ; can't load package 5
dsNoPk6 .EQU 23 ; can't load package 6
Inside Macintosh

dsNoPk7 .EQU 24 ;can't load package 7
dsMemFullErr .EQU 25 ;can't allocate requested block
dsBadLaunch .EQU 26 ;Segment Loader error
dsFSErr .EQU 27 ;file map destroyed
dsStkNHeap .EQU 28 ;stack overflow error
dsReinsert .EQU 30 ;"Please insert the disk:"
dsSysErr .EQU 32767 ;undifferentiated system error

Routines

Trap macro _SysError
On entry DO: errorCode (word)
On exit All registers changed

Variables

DSErrCode Current system error ID (word)
DSAlertTab Pointer to system error alert table in use
DSAlertRect Rectangle enclosing system error alert (8 bytes)

III-178 System Error Handler
TEXTEDIT

Constants

CONST \{ Text justification \}
    teJustLeft = 0;
    teJustCenter = 1;
    teJustRight = -1;

Data Types

TYPE TEHandle = "TEPtr;
    TEPtr = ^TERec;
    TERec = RECORD
        destRect: Rect; \{ destination rectangle \}
        viewRect: Rect; \{ view rectangle \}
        selRect: Rect; \{ used from assembly language \}
        lineHeight INTEGER; \{ for line spacing \}
        fontAscent: INTEGER; \{ caret/highlighting position \}
        selPoint: Point; \{ used from assembly language \}
        selStart: INTEGER; \{ start of selection range \}
        selEnd: INTEGER; \{ end of selection range \}
        active: INTEGER; \{ used internally \}
        wordBreak: ProcPtr; \{ for word break routine \}
        clickLoop: ProcPtr; \{ for click loop routine \}
        clickTime: LONGINT; \{ used internally \}
        clickLoc: INTEGER; \{ used internally \}
        caretTime: LONGINT; \{ used internally \}
        caretState: INTEGER; \{ used internally \}
        just: INTEGER; \{ justification of text \}
        teLength: INTEGER; \{ length of text \}
        hText: Handle; \{ text to be edited \}
        recallBack: INTEGER; \{ used internally \}
        recallLines: INTEGER; \{ used internally \}
        clikStuff: INTEGER; \{ used internally \}
        crOnly: INTEGER; \{ if <0, new line at Return only \}
        txFont: INTEGER; \{ text font \}
        txFace: Style; \{ character style \}
        txMode: INTEGER; \{ pen mode \}
        txSize: INTEGER; \{ font size \}
        inPort: GrafPtr; \{ grafPort \}
        highHook: ProcPtr; \{ used from assembly language \}
        caretHook: ProcPtr; \{ used from assembly language \}
        nLines: INTEGER; \{ number of lines \}
        lineStarts: ARRAY[0..16000] OF INTEGER \{ positions of line starts \}
    END;
CharsHandle = 'CharsPtr;
CharsPtr    = 'Chars;
Chars       = Packed Array [0..32000] of Char;

Routines

Initialization and Allocation

PROCEDURE TEInit;
FUNCTION TEnew (destRect, viewRect: Rect): TEHandle;
PROCEDURE TEdispose (hTE: TEHandle);

Accessing the Text of an Edit Record

PROCEDURE TESetText (text: Ptr; length: LONGINT; hTE: TEHandle);
FUNCTION TEGetText (hTE: TEHandle): CharsHandle;

Insertion Point and Selection Range

PROCEDURE TEIdle (hTE: TEHandle);
PROCEDURE TEClick (pt: Point; extend: BOOLEAN; hTE: TEHandle);
PROCEDURE TESetSelect (selStart, selEnd: LONGINT; hTE: TEHandle);
PROCEDURE TEAactivate (hTE: TEHandle);
PROCEDURE TEDeactivate (hTE: TEHandle);

Editing

PROCEDURE TEKey (key: CHAR; hTE: TEHandle);
PROCEDURE TECut (hTE: TEHandle);
PROCEDURE TECopy (hTE: TEHandle);
PROCEDURE TEPaste (hTE: TEHandle);
PROCEDURE TEDelete (hTE: TEHandle);
PROCEDURE TEInsert (text: Ptr; length: LONGINT; hTE: TEHandle);

Text Display and Scrolling

PROCEDURE TESetJust (just: INTEGER; hTE: TEHandle);
PROCEDURE TESupdate (rUpdate: Rect; hTE: TEHandle);
PROCEDURE TextBox (text: Ptr; length: LONGINT; box: Rect; just: INTEGER);
PROCEDURE TESScroll (dh, dv: INTEGER; hTE: TEHandle);
Scrap Handling [Not in ROM]

FUNCTION TEFromScraps : OSErr;
FUNCTION TEToScrap : OSErr;
FUNCTION TEScrapHandle : Handle;
FUNCTION TEGetScrapLen : LONGINT;
PROCEDURE TESetScrapLen (length: LONGINT);

Advanced Routines

PROCEDURE SetWordBreak (wBrkProc: ProcPtr; hTE: TEHandle); [Not in ROM]
PROCEDURE SetClikLoop (clikProc: ProcPtr; hTE: TEHandle); [Not in ROM]
PROCEDURE TECalText (hTE: TEHandle);

Word Break Routine

FUNCTION MyWordBreak (text: Ptr; charPos: INTEGER) : BOOLEAN;

Click Loop Routine

FUNCTION MyClikLoop : BOOLEAN;

Assembly-Language Information

Constants

; Text justification

tJustLeft .EQU 0
tJustCenter .EQU 1
tJustRight .EQU -1

Edit Record Data Structure

tDestRect Destination rectangle (8 bytes)
tViewRect View rectangle (8 bytes)
tSelRect Selection rectangle (8 bytes)
tLineHite For line spacing (word)
tAscent Caret/highlighting position (word)
tSelPoint Point selected with mouse (long)
tSelStart Start of selection range (word)
tSelEnd End of selection range (word)
tWordBreak Address of word break routine (see below)
tClikProc Address of click loop routine (see below)
tJust Justification of text (word)
Inside Macintosh

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>teLength</td>
<td>Length of text (word)</td>
</tr>
<tr>
<td>teTextH</td>
<td>Handle to text</td>
</tr>
<tr>
<td>teCROnly</td>
<td>If &lt;0, new line at Return only (byte)</td>
</tr>
<tr>
<td>teFont</td>
<td>Text font (word)</td>
</tr>
<tr>
<td>teFace</td>
<td>Character style (word)</td>
</tr>
<tr>
<td>teMode</td>
<td>Pen mode (word)</td>
</tr>
<tr>
<td>teSize</td>
<td>Font size (word)</td>
</tr>
<tr>
<td>teGrafPort</td>
<td>Pointer to grafPort</td>
</tr>
<tr>
<td>teHiHook</td>
<td>Address of text highlighting routine (see below)</td>
</tr>
<tr>
<td>teCarHook</td>
<td>Address of routine to draw caret (see below)</td>
</tr>
<tr>
<td>teNLines</td>
<td>Number of lines (word)</td>
</tr>
<tr>
<td>teLines</td>
<td>Positions of line starts (teNLines*2 bytes)</td>
</tr>
<tr>
<td>tcRecSize</td>
<td>Size in bytes of edit record except teLines field</td>
</tr>
</tbody>
</table>

Word break routine

On entry

A0: pointer to text
D0: character position (word)

On exit

Z condition code: 0 to break at specified character
1 not to break there

Click loop routine

On exit

D0: 1
D2: must be preserved

Text highlighting routine

On entry

A3: pointer to locked edit record

Caret drawing routine

On entry

A3: pointer to locked edit record

Variables

TEScrpHandle  Handle to TextEdit scrap
TEScrpLength  Size in bytes of TextEdit scrap (word)
TERecal       Address of routine to recalculate line starts (see below)
TEDoText      Address of multi-purpose routine (see below)

TERecal routine

On entry

A3: pointer to locked edit record
D7: change in length of edit record (word)

On exit

D2: line start of line containing first character to be redrawn (word)
D3: position of first character to be redrawn (word)
D4: position of last character to be redrawn (word)
TEDoText routine

On entry
A3: pointer to locked edit record
D3: position of first character to be redrawn (word)
D4: position of last character to be redrawn (word)
D7: (word) 0 to hit-test a character
     1 to highlight selection range
     -1 to display text
     -2 to position pen to draw caret

On exit
A0: pointer to current grafPort
D0: if hit-testing, character position or -1 for none (word)
UTILITIES, OPERATING SYSTEM

Constants

CONST { Values returned by Environ procedure }

macXLMachine = 0; {Macintosh XL}
macMachine = 1; {Macintosh 128K or 512K}

{ Result codes }

clkRdErr = -85; {unable to read clock}
clkWrErr = -86; {time written did not verify}
memFullErr = -108; {not enough room in heap zone}
memWZErr = -111; {attempt to operate on a free block}
niHandleErr = -109; {NIL master pointer}
nErr = 0; {no error}
prInitErr = -88; {validity status is not $A8}
prWrErr = -87; {parameter RAM written did not verify}
qErr = -1; {entry not in specified queue}

Data Types

TYPE OSType = PACKED ARRAY[1..4] OF CHAR;

OSErr = INTEGER;

SysPPtr = ^SysParmType;
SysParmType = RECORD
valid: Byte; {validity status}
aTalkA: Byte; {AppleTalk node ID hint for modem port}
aTalkB: Byte; {AppleTalk node ID hint for printer port}
config: Byte; {use types for serial ports}
portA: INTEGER; {modem port configuration}
portB: INTEGER; {printer port configuration}
alarm: LONGINT; {alarm setting}
font: INTEGER; {application font number minus 1}
kbdPrint: INTEGER; {auto-key settings, printer connection}
volClik: INTEGER; {speaker volume, double-click, caret blink}
misc: INTEGER {mouse scaling, startup disk, menu blink}
END;

QHdrPtr = ^QHdr;
QHdr = RECORD
qFlags: INTEGER; {queue flags}
qHead: QElemPtr; {first queue entry}
qTail: QElemPtr {last queue entry}
END;
QTypes = (dummyType, vType, ioQType, drvQType, evType, fsQType);
  {vertical retrace queue type}
  {file I/O or driver I/O queue type}
  {drive queue type}
  {event queue type}
  {volume-control-block queue type}

QElemPtr := QElem;
QElem = RECORD
  CASE QTypes OF
    vType: (vblQElem : VBLTask);
    ioQType: (ioQElem : ParamBlockRec);
    drvQType: (drvQElem : DrvQEl);
    evType: (evQElem : EvQEl);
    fsQType: (vcbQElem : VCB);
  END;

DateTimeRec = RECORD
  END;

FUNCTION HandToHand (VAR theHndl: Handle) : OSErr;
FUNCTION PtrToHand (srcPtr: Ptr; VAR dstHndl: Handle; size : LONGINT) : OSErr;
FUNCTION PtrToXHand (srcPtr: Ptr; dstHndl: Handle; size : LONGINT) : OSErr;
FUNCTION HandAndHand (aHndl,bHndl: Handle) : OSErr;
FUNCTION PtrAndHand (pntr: Ptr; hndl: Handle; size: LONGINT) : OSErr;

FUNCTION EqualString (aStr,bStr: Str255; caseSens, diacSens: BOOLEAN) : BOOLEAN;
PROCEDURE UprString (VAR theString: Str255; diacSens: BOOLEAN);
Inside Macintosh

Date and Time Operations

FUNCTION ReadDateTime (VAR secs: LONGINT) : OSErr;
PROCEDURE GetDateTime (VAR secs: LONGINT); [Not in ROM]
FUNCTION SetDateTime (secs: LONGINT) : OSErr;
PROCEDURE Date2Secs (date: DateTimeRec; VAR secs: LONGINT);
PROCEDURE Secs2Date (secs: LONGINT; VAR date: DateTimeRec);
PROCEDURE GetTime (VAR date: DateTimeRec); [Not in ROM]
PROCEDURE SetTime (date: DateTimeRec); [Not in ROM]

Parameter RAM Operations

FUNCTION InitUtil : OSErr;
FUNCTION GetSysPPtr : SysPtr; [Not in ROM]
FUNCTION WriteParam : OSErr;

Queue Manipulation

PROCEDURE Enqueue (qEntry: QElemPtr; theQueue: QHdrPtr);
FUNCTION Dequeue (qEntry: QElemPtr; theQueue: QHdrPtr) : OSErr;

Trap Dispatch Table Utilities

PROCEDURE SetTrapAddress (trapAddr: LONGINT; trapNum : INTEGER);
FUNCTION GetTrapAddress (trapNum: INTEGER) : LONGINT;

Miscellaneous Utilities

PROCEDURE Delay (numTicks: LONGINT; VAR finalTicks: LONGINT);
PROCEDURE SysBeep (duration: INTEGER);
PROCEDURE Environ (VAR rom, machine: INTEGER); [Not in ROM]
PROCEDURE Restart; [Not in ROM]
PROCEDURE SetUpA5; [Not in ROM]
PROCEDURE RestoreA5; [Not in ROM]

Default Parameter RAM Values

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Validity status</td>
<td>$A8</td>
</tr>
<tr>
<td>Node ID hint for modem port</td>
<td>0</td>
</tr>
<tr>
<td>Node ID hint for printer port</td>
<td>0</td>
</tr>
<tr>
<td>Use types for serial ports</td>
<td>0 (both ports)</td>
</tr>
<tr>
<td>Modem port configuration</td>
<td>9600 baud, 8 data bits, 2 stop bits, no parity</td>
</tr>
<tr>
<td>Parameter</td>
<td>Default value</td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td>--------------------------------------</td>
</tr>
<tr>
<td>Printer port configuration</td>
<td>Same as for modem port</td>
</tr>
<tr>
<td>Alarm setting</td>
<td>0 (midnight, January 1, 1904)</td>
</tr>
<tr>
<td>Application font number minus 1</td>
<td>2 (Geneva)</td>
</tr>
<tr>
<td>Auto-key threshold</td>
<td>6 (24 ticks)</td>
</tr>
<tr>
<td>Auto-key rate</td>
<td>3 (6 ticks)</td>
</tr>
<tr>
<td>Printer connection</td>
<td>0 (printer port)</td>
</tr>
<tr>
<td>Speaker volume</td>
<td>3 (medium)</td>
</tr>
<tr>
<td>Double-click time</td>
<td>8 (32 ticks)</td>
</tr>
<tr>
<td>Caret-blink time</td>
<td>8 (32 ticks)</td>
</tr>
<tr>
<td>Mouse scaling</td>
<td>1 (on)</td>
</tr>
<tr>
<td>Preferred system startup disk</td>
<td>0 (internal drive)</td>
</tr>
<tr>
<td>Menu blink</td>
<td>3</td>
</tr>
</tbody>
</table>

**Assembly-Language Information**

**Constants**

; Result codes

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Value</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>clkRdErr</td>
<td>-85</td>
<td>unable to read clock</td>
</tr>
<tr>
<td>clkWrErr</td>
<td>-86</td>
<td>time written did not verify</td>
</tr>
<tr>
<td>memFullErr</td>
<td>-108</td>
<td>not enough room in heap zone</td>
</tr>
<tr>
<td>memWZErr</td>
<td>-111</td>
<td>attempt to operate on a free block</td>
</tr>
<tr>
<td>nilHandleErr</td>
<td>-109</td>
<td>NIL master pointer</td>
</tr>
<tr>
<td>noErr</td>
<td>0</td>
<td>no error</td>
</tr>
<tr>
<td>prInitErr</td>
<td>-88</td>
<td>validity status is not $A8</td>
</tr>
<tr>
<td>prWrErr</td>
<td>-87</td>
<td>parameter RAM written did not verify</td>
</tr>
<tr>
<td>qErr</td>
<td>-1</td>
<td>entry not in specified queue</td>
</tr>
</tbody>
</table>

; Queue types

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Value</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>vType</td>
<td>1</td>
<td>vertical retrace queue type</td>
</tr>
<tr>
<td>ioQType</td>
<td>2</td>
<td>file I/O or driver I/O queue type</td>
</tr>
<tr>
<td>drvQType</td>
<td>3</td>
<td>drive queue type</td>
</tr>
<tr>
<td>evType</td>
<td>4</td>
<td>event queue type</td>
</tr>
<tr>
<td>fsQType</td>
<td>5</td>
<td>volume-control-block queue type</td>
</tr>
</tbody>
</table>

**Queue Data Structure**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>qFlags</td>
<td>Queue flags (word)</td>
</tr>
<tr>
<td>qHead</td>
<td>Pointer to first queue entry</td>
</tr>
<tr>
<td>qTail</td>
<td>Pointer to last queue entry</td>
</tr>
</tbody>
</table>
**Date/Time Record Data Structure**

- **dtYear**: 1904 to 2040 (word)
- **dtMonth**: 1 to 12 for January to December (word)
- **dtDay**: 1 to 31 (word)
- **dtHour**: 0 to 23 (word)
- **dtMinute**: 0 to 59 (word)
- **dtSecond**: 0 to 59 (word)
- **dtDayOfWeek**: 1 to 7 for Sunday to Saturday (word)

**Routines**

<table>
<thead>
<tr>
<th>Routine</th>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_HandToHand</td>
<td>A0: theHndl (handle)</td>
<td>On entry: A0: srcPtr (ptr), D0: size (long)</td>
</tr>
<tr>
<td>_PtrToHand</td>
<td>A0: srcPtr (ptr)</td>
<td>On exit: A0: dstHndl (handle), D0: result code (word)</td>
</tr>
<tr>
<td>_PtrToXHand</td>
<td>A0: srcPtr (ptr), A1: dstHndl (handle), D0: size (long)</td>
<td></td>
</tr>
<tr>
<td>_HandAndHand</td>
<td>A0: aHndl (handle), A1: bHndl (handle), D0: size (long)</td>
<td></td>
</tr>
<tr>
<td>_CmpString</td>
<td>A0: ptr to first string, A1: ptr to second string, D0: high word: length of first string, low word: length of second string</td>
<td>On exit: A0: bHndl (handle), D0: result code (word)</td>
</tr>
<tr>
<td>_UprString</td>
<td>A0: ptr to string</td>
<td>A0: ptr to string, D0: length of string (word)</td>
</tr>
<tr>
<td>_ReadDateTime</td>
<td>A0: ptr to long word secs</td>
<td>A0: ptr to long word secs, D0: result code (word)</td>
</tr>
<tr>
<td>_SetDateTime</td>
<td>D0: secs (long)</td>
<td>D0: result code (word)</td>
</tr>
<tr>
<td>_Date2Secs</td>
<td>A0: ptr to date/time record</td>
<td>D0: secs (long)</td>
</tr>
<tr>
<td>_Secs2Date</td>
<td>D0: secs (long)</td>
<td>A0: ptr to date/time record</td>
</tr>
<tr>
<td>_InitUtil</td>
<td>A0: SysParam (ptr), D0: MinusOne (long)</td>
<td>D0: result code (word)</td>
</tr>
</tbody>
</table>

**III-188 Utilities, Operating System**
Trap macro

_onEntry

A0: qEntry (ptr)
A1: theQueue (ptr)

_onDequeue

A0: qEntry (ptr)
A1: theQueue (ptr)

_getTrapAddress

D0: trapNum (word)

_setTrapAddress

A0: trapAddr (address)
D0: trapNum (word)

_Delay

A0: numTicks (long)
D0: finalTicks (long)

_SysBeep

stack: duration (word)

Variables

SysParam        Low-memory copy of parameter RAM (20 bytes)
SPValid         Validity status (byte)
SPATalkA        AppleTalk node ID hint for modem port (byte)
SPATalkB        AppleTalk node ID hint for printer port (byte)
SPConfig        Use types for serial ports (byte)
SPPortA         Modem port configuration (word)
SPPortB         Printer port configuration (word)
SPAlarm         Alarm setting (long)
SPFont          Application font number minus 1 (word)
SPKbd           Auto-key threshold and rate (byte)
SPPrint         Printer connection (byte)
SPVolCtI        Speaker volume (byte)
SPClikCaret     Double-click and caret-blink times (byte)
SPMisc2         Mouse scaling, system startup disk, menu blink (byte)
CrsrThresh      Mouse-scaling threshold (word)
Time            Seconds since midnight, January 1, 1904 (long)
UTILITIES, TOOLBOX

Constants

CONST { Resource ID of standard pattern list }
  sysPatListID = 0;

{ Resource IDs of standard cursors }
  iBeamCursor = 1; {to select text }
  crossCursor = 2; {to draw graphics}
  plusCursor = 3; {to select cells in structured documents}
  watchCursor = 4; {to indicate a long wait}

Data Types

TYPE Int64Bit = RECORD
  hiLong: LONGINT;
  loLong: LONGINT
END;

CursPtr = ^Cursor;
CursHandle = ^CursPtr;

PatPtr = ^Pattern;
PatHandle = ^PatPtr;

Routines

Fixed-Point Arithmetic

FUNCTION FixRatio (numer, denom: INTEGER) : Fixed;
FUNCTION FixMul (a, b: Fixed) : Fixed;
FUNCTION FixRound (x: Fixed) : INTEGER;

String Manipulation

FUNCTION NewString (theString: Str255) : StringHandle;
PROCEDURE SetString (h: StringHandle; theString: Str255);
FUNCTION GetString (stringID: INTEGER) : StringHandle;
PROCEDURE GetIndString (VAR theString: Str255; strListID: INTEGER;
  index: INTEGER); [Not in ROM]
Byte Manipulation

FUNCTION Munger (h: Handle; offset: LONGINT; ptr1: Ptr; len1: LONGINT; ptr2: Ptr; len2: LONGINT) : LONGINT;
PROCEDURE PackBits (VAR srcPtr,dstPtr: Ptr; srcBytes: INTEGER);
PROCEDURE UnpackBits (VAR srcPtr,dstPtr: Ptr; dstBytes: INTEGER);

Bit Manipulation

FUNCTION BitTst (bytePtr: Ptr; bitNum: LONGINT) : BOOLEAN;
PROCEDURE BitSet (bytePtr: Ptr; bitNum: LONGINT);
PROCEDURE BitClr (bytePtr: Ptr; bitNum: LONGINT);

Logical Operations

FUNCTION BitAnd (value1,value2: LONGINT) : LONGINT;
FUNCTION BitOr (value1,value2: LONGINT) : LONGINT;
FUNCTION BitXor (value1,value2: LONGINT) : LONGINT;
FUNCTION BitNot (value: LONGINT) : LONGINT;
FUNCTION BitShift (value: LONGINT; count: INTEGER) : LONGINT;

Other Operations on Long Integers

FUNCTION HiWord (x: LONGINT) : INTEGER;
FUNCTION LoWord (x: LONGINT) : INTEGER;
PROCEDURE LongMul (a,b : LONGINT; VAR dest: Int64Bit);

Graphics Utilities

PROCEDURE ScreenRes (VAR scrnHRes, scrnVRes: INTEGER); [Not in ROM]
FUNCTION GetIcon (iconID: INTEGER) : Handle;
PROCEDURE PlotIcon (theRect: Rect; theIcon: Handle);
FUNCTION GetPattern (patID: INTEGER) : PatHandle;
PROCEDURE GetIndPattern (VAR thePattern: Pattern; patListID: INTEGER; index: INTEGER); [Not in ROM]
FUNCTION GetCursor (cursorID: INTEGER) : CursHandle;
PROCEDURE ShieldCursor (shieldRect: Rect; offsetPt: Point);
FUNCTION GetPicture (picID: INTEGER) : PicHandle;

Miscellaneous Utilities

FUNCTION DeltaPoint (ptA,ptB: Point) : LONGINT;
FUNCTION SlopeFromAngle (angle: INTEGER) : Fixed;
FUNCTION AngleFromSlope (slope: Fixed) : INTEGER;
Assembly-Language Information

Constants

; Resource ID of standard pattern list
sysPatListID .EQU 0

; Resource IDs of standard cursors
iBeamCursor .EQU 1 ;to select text
crossCursor .EQU 2 ;to draw graphics
plusCursor .EQU 3 ;to select cells in structured documents
watchCursor .EQU 4 ;to indicate a long wait

Variables

ScrVRes Pixels per inch vertically (word)
ScrHRes Pixels per inch horizontally (word)
VERTICAL RETRACE MANAGER

Constants

CONST { Result codes }

noErr = 0;  {no error}
qErr = -1;  {task entry isn't in the queue}
vTypErr = -2;  {qType field isn't ORD(vType)}

Data Types

TYPE VBLTask = RECORD
  qLink: QElemPtr;  {next queue entry}
  qType: INTEGER;  {queue type}
  vblAddr: ProcPtr;  {pointer to task}
  vblCount: INTEGER;  {task frequency}
  vblPhase: INTEGER  {task phase}
END;

Routines

FUNCTION VInstall (vblTaskPtr: QElemPtr) : OSErr;
FUNCTION VRemove (vblTaskPtr: QElemPtr) : OSErr;
FUNCTION GetVBLQHdr : QHdrPtr;  [Not in ROM]

Assembly-Language Information

Constants

inVBL .EQU 6  ;set if Vertical Retrace Manager is executing a task

; Result codes

noErr .EQU 0  ;no error
qErr .EQU -1  ;task entry isn't in the queue
vTypErr .EQU -2  ;qType field isn't vType

Structure of Vertical Retrace Queue Entry

qLink          Pointer to next queue entry
qType          Queue type (word)
vblAddr        Address of task
vblCount       Task frequency (word)
vblPhase       Task phase (word)
Routines

<table>
<thead>
<tr>
<th>Trap macro</th>
<th>On entry</th>
<th>On exit</th>
</tr>
</thead>
<tbody>
<tr>
<td>_VInstall</td>
<td>A0: vblTaskPtr (ptr)</td>
<td>D0: result code (word)</td>
</tr>
<tr>
<td>_VRemove</td>
<td>A0: vblTaskPtr (ptr)</td>
<td>D0: result code (word)</td>
</tr>
</tbody>
</table>

Variables

VBLQueue Vertical retrace queue header (10 bytes)
WINDOW MANAGER

**Constants**

```plaintext
CONST [ Window definition IDs ]

documentProc = 0;  [standard document window]
dBoxProc    = 1;  [alert box or modal dialog box]
plainDBox   = 2;  [plain box]
altDBoxProc = 3;  [plain box with shadow]
ocGrowDocProc = 4;  [document window without size box]
rDocProc    = 16;  [rounded-corner window]

[ Window class, in windowKind field of window record ]
dialogKind = 2;  [dialog or alert window]
userKind = 8;  [window created directly by the application]

[ Values returned by FindWindow ]
inDesk = 0;  [none of the following]
inMenuBar = 1;  [in menu bar]
inSysWindow = 2;  [in system window]
inContent = 3;  [in content region (except grow, if active)]
inDrag = 4;  [in drag region]
inGrow = 5;  [in grow region (active window only)]
inGoAway = 6;  [in go-away region (active window only)]

[ Axis constraints for DragGrayRgn ]
noConstraint = 0;  [no constraint]
hAxisOnly = 1;  [horizontal axis only]
vAxisOnly = 2;  [vertical axis only]

[ Messages to window definition function ]
wDraw = 0;  [draw window frame]
wHit = 1;  [tell what region mouse button was pressed in]
wCalcRgns = 2;  [calculate strucRgn and contRgn]
wNew = 3;  [do any additional window initialization]
wDispose = 4;  [take any additional disposal actions]
wGrow = 5;  [draw window's grow image]
wDrawGIcon = 6;  [draw size box in content region]

[ Values returned by window definition function's hit routine ]
wNoHit = 0;  [none of the following]
wInContent = 1;  [in content region (except grow, if active)]
wInDrag = 2;  [in drag region]
wInGrow = 3;  [in grow region (active window only)]
wInGoAway = 4;  [in go-away region (active window only)]
```
Data Types

TYPE WindowPtr = GrafPtr;
WindowPeek = ^WindowRecord;

WindowRecord =
RECORD
  port: GrafPort; {window's grafPort}
  windowKind: INTEGER; {window class}
  visible: BOOLEAN; {TRUE if visible}
  hilited: BOOLEAN; {TRUE if highlighted}
  goAwayFlag: BOOLEAN; {TRUE if has go-away region}
  spareFlag: BOOLEAN; {reserved for future use}
  strucRgn: RgnHandle; {structure region}
  contRgn: RgnHandle; {content region}
  updateRgn: RgnHandle; {update region}
  windowDefProc: Handle; {window definition function}
  dataHandle: Handle; {data used by windowDefProc}
  titleHandle: StringHandle; {window's title}
  titleWidth: INTEGER; {width of title in pixels}
  controlList: ControlHandle; {window's control list}
  nextWindow: WindowPeek; {next window in window list}
  windowPic: PicHandle; {picture for drawing window}
  refCon: LONGINT {window's reference value}
END;

Routines

Initialization and Allocation

PROCEDURE InitWindows;
PROCEDURE GetWMgrPort (VAR wPort: GrafPtr);
FUNCTION NewWindow (wStorage: Ptr; boundsRect: Rect; title: Str255;
  visible: BOOLEAN; procID: INTEGER; behind: WindowPtr;
  goAwayFlag: BOOLEAN; refCon: LONGINT) : WindowPtr;
FUNCTION GetNewWindow (windowID: INTEGER; wStorage: Ptr; behind:
  WindowPtr) : WindowPtr;
PROCEDURE CloseWindow (theWindow: WindowPtr);
PROCEDURE DisposeWindow (theWindow: WindowPtr);
Window Display

PROCEDURE SetWTitle (theWindow: WindowPtr; title: Str255);
PROCEDURE GetWTitle (theWindow: WindowPtr; VAR title: Str255);
PROCEDURE SelectWindow (theWindow: WindowPtr);
PROCEDURE HideWindow (theWindow: WindowPtr);
PROCEDURE ShowWindow (theWindow: WindowPtr);
PROCEDURE ShowHide (theWindow: WindowPtr; showFlag: BOOLEAN);
PROCEDURE HiliteWindow (theWindow: WindowPtr; fHilite: BOOLEAN);
PROCEDURE BringToFront (theWindow: WindowPtr);
PROCEDURE SendBehind (theWindow, behindWindow: WindowPtr);
FUNCTION FrontWindow: WindowPtr;
PROCEDURE DrawGrowIcon (theWindow: WindowPtr);

Mouse Location

FUNCTION FindWindow (thePt: Point; VAR whichWindow: WindowPtr) : INTEGER;
FUNCTION TrackGoAway (theWindow: WindowPtr; thePt: Point) : BOOLEAN;

Window Movement and Sizing

PROCEDURE MoveWindow (theWindow: WindowPtr; hGlobal, vGlobal: INTEGER; front: BOOLEAN);
PROCEDURE DragWindow (theWindow: WindowPtr; startPt: Point; boundsRect: Rect);
FUNCTION GrowWindow (theWindow: WindowPtr; startPt: Point; sizeRect: Rect) : LONGINT;
PROCEDURE SizeWindow (theWindow: WindowPtr; w, h: INTEGER; fUpdate: BOOLEAN);

Update Region Maintenance

PROCEDURE InvalRect (badRect: Rect);
PROCEDURE InvalRgn (badRgn: RgnHandle);
PROCEDURE ValidRect (goodRect: Rect);
PROCEDURE ValidRgn (goodRgn: RgnHandle);
PROCEDURE BeginUpdate (theWindow: WindowPtr);
PROCEDURE EndUpdate (theWindow: WindowPtr);

Miscellaneous Routines

PROCEDURE SetWRefCon (theWindow: WindowPtr; data: LONGINT);
FUNCTION GetWRefCon (theWindow: WindowPtr) : LONGINT;
PROCEDURE SetWindowPic (theWindow: WindowPtr; pic: PicHandle);
FUNCTION GetWindowPic (theWindow: WindowPtr) : PicHandle;
FUNCTION PinRect (theRect: Rect; thePt: Point) : LONGINT;
FUNCTION DragGrayRgn (theRgn: RgnHandle; startPt: Point; limitRect, slopRect: Rect; axis: INTEGER; actionProc: ProcPtr) : LONGINT;

Low-Level Routines

FUNCTION CheckUpdate (VAR theEvent: EventRecord) : BOOLEAN;
PROCEDURE ClipAbove (window: WindowPeek);
PROCEDURE SaveOld (window: WindowPeek);
PROCEDURE DrawNew (window: WindowPeek; update: BOOLEAN);
PROCEDURE PaintOne (window: WindowPeek; clobberedRgn: RgnHandle);
PROCEDURE PaintBehind (startWindow: WindowPeek; clobberedRgn: RgnHandle);
PROCEDURE CalcVis (window: WindowPeek);
PROCEDURE CalcVisBehind (startWindow: WindowPeek; clobberedRgn: RgnHandle);

Diameters of Curvature for Rounded-Corner Windows

<table>
<thead>
<tr>
<th>Window definition ID</th>
<th>Diameters of curvature</th>
</tr>
</thead>
<tbody>
<tr>
<td>rDocProc</td>
<td>16, 16</td>
</tr>
<tr>
<td>rDocProc + 1</td>
<td>4, 4</td>
</tr>
<tr>
<td>rDocProc + 2</td>
<td>6, 6</td>
</tr>
<tr>
<td>rDocProc + 3</td>
<td>8, 8</td>
</tr>
<tr>
<td>rDocProc + 4</td>
<td>10, 10</td>
</tr>
<tr>
<td>rDocProc + 5</td>
<td>12, 12</td>
</tr>
<tr>
<td>rDocProc + 6</td>
<td>20, 20</td>
</tr>
<tr>
<td>rDocProc + 7</td>
<td>24, 24</td>
</tr>
</tbody>
</table>

Window Definition Function

FUNCTION MyWindow (varCode: INTEGER; theWindow: WindowPtr; message: INTEGER; param: LONGINT) : LONGINT;

Assembly-Language Information

Constants

; Window definition IDs

documentProc .EQU 0 ;standard document window
dBoxProc .EQU 1 ;alert box or modal dialog box
plainDBox .EQU 2 ;plain box
altDBoxProc .EQU 3 ;plain box with shadow
noGrowDocProc .EQU 4 ;document window without size box
rDocProc .EQU 16 ;rounded-corner window

; Window class, in windowKind field of window record
dialogKind .EQU 2 ;dialog or alert window
userKind .EQU 8 ;window created directly by the application

; Values returned by FindWindow
inDesk .EQU 0 ;none of the following
inMenuBar .EQU 1 ;in menu bar
inSysWindow .EQU 2 ;in system window
inContent .EQU 3 ;in content region (except grow, if active)
inDrag .EQU 4 ;in drag region
inGrow .EQU 5 ;in grow region (active window only)
inGoAway .EQU 6 ;in go-away region (active window only)

; Axis constraints for DragGrayRgn
noConstraint .EQU 0 ;no constraint
hAxisOnly .EQU 1 ;horizontal axis only
vAxisOnly .EQU 2 ;vertical axis only

; Messages to window definition function
wDrawMsg .EQU 0 ;draw window frame
wHitMsg .EQU 1 ;tell what region mouse button was pressed in
wCalcRgnMsg .EQU 2 ;calculate strucRgn and contRgn
wInitMsg .EQU 3 ;do any additional window initialization
wDisposeMsg .EQU 4 ;take any additional disposal actions
wGrowMsg .EQU 5 ;draw window's grow image
wGIconMsg .EQU 6 ;draw size box in content region

; Value returned by window definition function's hit routine
wNoHit .EQU 0 ;none of the following
wInContent .EQU 1 ;in content region (except grow, if active)
wInDrag .EQU 2 ;in drag region
wInGrow .EQU 3 ;in grow region (active window only)
wInGoAway .EQU 4 ;in go-away region (active window only)

; Resource ID of desktop pattern
deskPatID .EQU 16

Window Record Data Structure

windowPort Window's grafPort (portRec bytes)
windowKind Window class (word)
wVisible Nonzero if window is visible (byte)
wHilited Nonzero if window is highlighted (byte)
Inside Macintosh

wGoAway  Nonzero if window has go-away region (byte)
structRgn  Handle to structure region of window
contRgn  Handle to content region of window
updateRgn  Handle to update region of window
windowDef  Handle to window definition function
wDataHandle  Handle to data used by window definition function
wTitleHandle  Handle to window's title (preceded by length byte)
wTitleWidth  Width of title in pixels (word)
wControlList  Handle to window's control list
nextWindow  Pointer to next window in window list
windowPic  Picture handle for drawing window
wRefCon  Window's reference value (long)
windowSize  Size in bytes of window record

Special Macro Names

Pascal name  Macro name
CalcVisBehind  _CalcVBehind
DisposeWindow  _DisposWindow
DragGrayRgn  _DragGrayRgn or, after setting the global variable DragPattern,
             _DragTheRgn

Variables

WindowList  Pointer to first window in window list
SaveUpdate  Flag for whether to generate update events (word)
PaintWhite  Flag for whether to paint window white before update event (word)
CurActivate  Pointer to window to receive activate event
CurDeactive  Pointer to window to receive deactivate event
GrayRgn  Handle to region drawn as desktop
DeskPattern  Pattern with which desktop is painted (8 bytes)
DeskHook  Address of procedure for painting desktop or responding to clicks on desktop
WMgrPort  Pointer to Window Manager port
GhostWindow  Pointer to window never to be considered frontmost
DragHook  Address of procedure to execute during TrackGoAway, DragWindow,
           GrowWindow, and DragGrayRgn
DragPattern  Pattern of dragged region's outline (8 bytes)
OldStructure  Handle to saved structure region
OldContent  Handle to saved content region
SaveVisRgn  Handle to saved visRgn
ASSEMBLY LANGUAGE

Variables

OneOne $00010001
MinusOne $FFFFFFFE
Lo3Bytes $00FFFFFF
Scratch20 20-byte scratch area
Scratch8 8-byte scratch area
Too1Scratch 8-byte scratch area
ApplScratch 12-byte scratch area reserved for use by applications
ROMBase Base address of ROM
RAMBase Trap dispatch table's base address for routines in RAM
CurrentA5 Address of boundary between application globals and application parameters

Hardware

Warning: This information applies only to the Macintosh 128K and 512K, not to the Macintosh XL.

Constants

; VIA base addresses
vBase .EQU $EFE1FE ;main base for VIA chip (in variable VIA)
aVBufB .EQU vBase ;register B base
aVBufA .EQU $EFFFFE ;register A base
aVBufM .EQU aVBufB ;register containing mouse signals
aVIFR .EQU $EFFBFE ;interrupt flag register
aVIER .EQU $EFFDFE ;interrupt enable register

; Offsets from vBase
vBufB .EQU 512*0 ;register B (zero offset)
vDirB .EQU 512*2 ;register B direction register
vDirA .EQU 512*3 ;register A direction register
vT1C .EQU 512*4 ;timer 1 counter (low-order byte)
vT1CH .EQU 512*5 ;timer 1 counter (high-order byte)
vTIL .EQU 512*6 ;timer 1 latch (low-order byte)
vT1LH .EQU 512*7 ;timer 1 latch (high-order byte)
vT2C .EQU 512*8 ;timer 2 counter (low-order byte)
vT2CH .EQU 512*9 ;timer 2 counter (high-order byte)
vSR .EQU 512*10 ;shift register (keyboard)
vACR .EQU 512*11 ;auxiliary control register
vPCR .EQU 512*12 ;peripheral control register
vIFR .EQU 512*13 ;interrupt flag register
; VIA register A constants
vAOut .EQU $7F ; direction register A: 1 bits = outputs
vAInit .EQU $7B ; initial value for vBufA (medium volume)
vSound .EQU 7 ; sound volume bits

; VIA register A bit numbers
vSndPg2 .EQU 3 ; 0 = alternate sound buffer
vOverlay .EQU 4 ; 1 = ROM overlay (system startup only)
vHeadSel .EQU 5 ; disk SEL control line
vPage2 .EQU 6 ; 0 = alternate screen buffer
vSCCWReq .EQU 7 ; SCC wait/request line

; VIA register B constants
vBOut .EQU $87 ; direction register B: 1 bits = outputs
vBInit .EQU $07 ; initial value for vBufB

; VIA register B bit numbers
rTCDta .EQU 0 ; real-time clock serial data line
rTCClk .EQU 1 ; real-time clock data-clock line
rTCEnb .EQU 2 ; real-time clock serial enable
vSW .EQU 3 ; 0 = mouse button is down
vX2 .EQU 4 ; mouse X quadrature level
vY2 .EQU 5 ; mouse Y quadrature level
vH4 .EQU 6 ; 1 = horizontal blanking
vSndEnb .EQU 7 ; 0 = sound enabled, 1 = disabled

; SCC base addresses
sccRBase .EQU $9FFFF8 ; SCC base read address (in variable SCCRd)
sccWBase .EQU $BFFFF9 ; SCC base write address (in variable SCCWr)

; Offsets from SCC base addresses
aData .EQU 6 ; channel A data in or out
aCtl .EQU 2 ; channel A control
bData .EQU 4 ; channel B data in or out
bCtl .EQU 0 ; channel B control

; Bit numbers for control register RR0
rxBF .EQU 0 ; 1 = SCC receive buffer full
txBE .EQU 2 ; 1 = SCC send buffer empty
; IWM base address

ibase .EQU $DFE1FF ;IWM base address (in variable IWM)

; Offsets from ibase

ph0L  .EQU 512*0  ;CA0 off (0)
ph0H  .EQU 512*1  ;CA0 on (1)
ph1L  .EQU 512*2  ;CA1 off (0)
ph1H  .EQU 512*3  ;CA1 on (1)
ph2L  .EQU 512*4  ;CA2 off (0)
ph2H  .EQU 512*5  ;CA2 on (1)
ph3L  .EQU 512*6  ;LSTRB off (low)
ph3H  .EQU 512*7  ;LSTRB on (high)
mtrg  .EQU 512*8  ;disk enable off
mtnOn  .EQU 512*9  ;disk enable on
intDrive .EQU 512*10 ;select internal drive
extDrive .EQU 512*11 ;select external drive
q2L  .EQU 512*12  ;Q6 off
q2H  .EQU 512*13  ;Q6 on
q7L  .EQU 512*14  ;Q7 off
q7H  .EQU 512*15  ;Q7 on

; Screen and sound addresses for 512K Macintosh (will also work for
; 128K, since addresses wrap)

screenLow .EQU $7A700 ;top left corner of main screen buffer
soundLow .EQU $7FD00 ;main sound buffer (in variable SoundBase)
pwmBuffer .EQU $7FD01 ;main disk speed buffer
ovlyRAM .EQU $600000 ;RAM start address when overlay is set
ovlyScreen .EQU $67A700 ;screen start with overlay set
romStart .EQU $400000 ;ROM start address (in variable ROMBase)

Variables

ROMBase  Base address of ROM
SoundBase Address of main sound buffer
SCCRd  SCC read base address
SCCWr  SCC write base address
IWM  IWM base address
VIA  VIA base address

Exception Vectors

<table>
<thead>
<tr>
<th>Location</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>$00</td>
<td>Reset: initial stack pointer (not a vector)</td>
</tr>
<tr>
<td>$04</td>
<td>Reset: initial vector</td>
</tr>
<tr>
<td>$08</td>
<td>Bus error</td>
</tr>
<tr>
<td>Location</td>
<td>Purpose</td>
</tr>
<tr>
<td>----------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>$0C</td>
<td>Address error</td>
</tr>
<tr>
<td>$10</td>
<td>Illegal instruction</td>
</tr>
<tr>
<td>$14</td>
<td>Divide by zero</td>
</tr>
<tr>
<td>$18</td>
<td>CHK instruction</td>
</tr>
<tr>
<td>$1C</td>
<td>TRAPV instruction</td>
</tr>
<tr>
<td>$20</td>
<td>Privilege violation</td>
</tr>
<tr>
<td>$24</td>
<td>Trace interrupt</td>
</tr>
<tr>
<td>$28</td>
<td>Line 1010 emulator</td>
</tr>
<tr>
<td>$2C</td>
<td>Line 1111 emulator</td>
</tr>
<tr>
<td>$30-$3B</td>
<td>Unassigned (reserved)</td>
</tr>
<tr>
<td>$3C</td>
<td>Uninitialized interrupt</td>
</tr>
<tr>
<td>$40-$5F</td>
<td>Unassigned (reserved)</td>
</tr>
<tr>
<td>$60</td>
<td>Spurious interrupt</td>
</tr>
<tr>
<td>$64</td>
<td>VIA interrupt</td>
</tr>
<tr>
<td>$68</td>
<td>SCC interrupt</td>
</tr>
<tr>
<td>$6C</td>
<td>VIA+SCC vector (temporary)</td>
</tr>
<tr>
<td>$70</td>
<td>Interrupt switch</td>
</tr>
<tr>
<td>$74</td>
<td>Interrupt switch + VIA</td>
</tr>
<tr>
<td>$78</td>
<td>Interrupt switch + SCC</td>
</tr>
<tr>
<td>$7C</td>
<td>Interrupt switch + VIA + SCC</td>
</tr>
<tr>
<td>$80-$BF</td>
<td>TRAP instructions</td>
</tr>
<tr>
<td>$C0-$FF</td>
<td>Unassigned (reserved)</td>
</tr>
</tbody>
</table>
APPENDIX A: RESULT CODES

This appendix lists all the result codes returned by the Macintosh system software. They're ordered by value, for convenience when debugging; the names you should actually use in your program are also listed.

The result codes are grouped roughly according to the lowest level at which the error may occur. This doesn't mean that only routines at that level may cause those errors; higher-level software may yield the same result codes. For example, an Operating System Utility routine that calls the Memory Manager may return one of the Memory Manager result codes. Where a different or more specific meaning is appropriate in a different context, that meaning is also listed.

<table>
<thead>
<tr>
<th>Value</th>
<th>Name</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>noErr</td>
<td>No error</td>
</tr>
</tbody>
</table>

Operating System Event Manager Error

<table>
<thead>
<tr>
<th>Value</th>
<th>Name</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>evtNotEnb</td>
<td>Event type not designated in system event mask</td>
</tr>
</tbody>
</table>

Printing Manager Errors

<table>
<thead>
<tr>
<th>Value</th>
<th>Name</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>128</td>
<td>iPrAbort</td>
<td>Application or user requested abort</td>
</tr>
<tr>
<td>-1</td>
<td>iPrSavPFil</td>
<td>Saving spool file</td>
</tr>
</tbody>
</table>

Queuing Errors

<table>
<thead>
<tr>
<th>Value</th>
<th>Name</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>qErr</td>
<td>Entry not in queue</td>
</tr>
<tr>
<td>-2</td>
<td>vTypErr</td>
<td>QType field of entry in vertical retrace queue isn't vType (in Pascal, ORD(vType))</td>
</tr>
</tbody>
</table>

Device Manager Errors

<table>
<thead>
<tr>
<th>Value</th>
<th>Name</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>-17</td>
<td>controlErr</td>
<td>Driver can't respond to this Control call</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Unimplemented control instruction (Printing Manager)</td>
</tr>
<tr>
<td>-18</td>
<td>statusErr</td>
<td>Driver can't respond to this Status call</td>
</tr>
<tr>
<td>-19</td>
<td>readErr</td>
<td>Driver can't respond to Read calls</td>
</tr>
<tr>
<td>-20</td>
<td>writErr</td>
<td>Driver can't respond to Write calls</td>
</tr>
<tr>
<td>-21</td>
<td>badUnitErr</td>
<td>Driver reference number doesn't match unit table</td>
</tr>
<tr>
<td>-22</td>
<td>unitEmptyErr</td>
<td>Driver reference number specifies NIL handle in unit table</td>
</tr>
<tr>
<td>-23</td>
<td>openErr</td>
<td>Requested read/write permission doesn't match driver's open permission</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Attempt to open RAM Serial Driver failed</td>
</tr>
<tr>
<td>-25</td>
<td>dRemovErr</td>
<td>Attempt to remove an open driver</td>
</tr>
<tr>
<td>-26</td>
<td>dlInstErr</td>
<td>Couldn't find driver in resource file</td>
</tr>
</tbody>
</table>

Result Codes III-205
### Inside Macintosh

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>-27</td>
<td>abortErr</td>
<td>I/O request aborted by KillIO</td>
</tr>
<tr>
<td></td>
<td>iIOAbort</td>
<td>I/O abort error (Printing Manager)</td>
</tr>
<tr>
<td>-28</td>
<td>notOpenErr</td>
<td>Driver isn't open</td>
</tr>
</tbody>
</table>

#### File Manager Errors

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>-33</td>
<td>dirFulErr</td>
<td>File directory full</td>
</tr>
<tr>
<td>-34</td>
<td>dskFulErr</td>
<td>All allocation blocks on the volume are full</td>
</tr>
<tr>
<td>-35</td>
<td>nsvErr</td>
<td>Specified volume doesn’t exist</td>
</tr>
<tr>
<td>-36</td>
<td>ioErr</td>
<td>I/O error</td>
</tr>
<tr>
<td>-37</td>
<td>bdNamErr</td>
<td>Bad file name or volume name (perhaps zero-length)</td>
</tr>
<tr>
<td>-38</td>
<td>fnOpnErr</td>
<td>File not open</td>
</tr>
<tr>
<td>-39</td>
<td>eofErr</td>
<td>Logical end-of-file reached during read operation</td>
</tr>
<tr>
<td>-40</td>
<td>posErr</td>
<td>Attempt to position before start of file</td>
</tr>
<tr>
<td>-42</td>
<td>tmfoErr</td>
<td>Too many files open</td>
</tr>
<tr>
<td>-43</td>
<td>fnfErr</td>
<td>File not found</td>
</tr>
<tr>
<td>-44</td>
<td>wPrErr</td>
<td>Volume is locked by a hardware setting</td>
</tr>
<tr>
<td>-45</td>
<td>fLckdErr</td>
<td>File is locked</td>
</tr>
<tr>
<td>-46</td>
<td>vLckdErr</td>
<td>Volume is locked by a software flag</td>
</tr>
<tr>
<td>-47</td>
<td>fBsyErr</td>
<td>File is busy; one or more files are open</td>
</tr>
<tr>
<td>-48</td>
<td>dupFNErr</td>
<td>File with specified name and version number already exists</td>
</tr>
<tr>
<td>-49</td>
<td>opWrErr</td>
<td>The read/write permission of only one access path to a file can allow writing</td>
</tr>
<tr>
<td>-50</td>
<td>paramErr</td>
<td>Error in parameter list</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Parameters don't specify an existing volume, and there's no default volume (File Manager)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bad positioning information (Disk Driver)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bad drive number (Disk Initialization Package)</td>
</tr>
<tr>
<td>-51</td>
<td>rfNumErr</td>
<td>Path reference number specifies nonexistent access path</td>
</tr>
<tr>
<td>-52</td>
<td>gfpErr</td>
<td>Error during GetFPos</td>
</tr>
<tr>
<td>-53</td>
<td>volOffLinErr</td>
<td>Volume not on-line</td>
</tr>
<tr>
<td>-54</td>
<td>permErr</td>
<td>Attempt to open locked file for writing</td>
</tr>
<tr>
<td>-55</td>
<td>volOnLinErr</td>
<td>Specified volume is already mounted and on-line</td>
</tr>
<tr>
<td>-56</td>
<td>nsDrvErr</td>
<td>No such drive; specified drive number doesn’t match any number in the drive queue</td>
</tr>
<tr>
<td>-57</td>
<td>noMacDskErr</td>
<td>Not a Macintosh disk; volume lacks Macintosh-format directory</td>
</tr>
<tr>
<td>-58</td>
<td>extFSErr</td>
<td>External file system; file-system identifier is nonzero, or path reference number is greater than 1024</td>
</tr>
</tbody>
</table>
Result Codes

-59 fsRnErr Problem during rename
-60 badMDBErr Bad master directory block; must reinitialize volume
-61 wrPermErr Read/write permission doesn't allow writing

Low-Level Disk Errors

-64 noDriveErr Drive isn't connected
-65 offLinErr No disk in drive
-66 noNybErr Disk is probably blank
-67 noAdrMkErr Can't find an address mark
-68 dataVerErr Read-verify failed
-69 badCksumErr Bad address mark
-70 badBtSlpErr Bad address mark
-71 noDtaMkErr Can't find a data mark
-72 badDCksum Bad data mark
-73 badDBtSlp Bad data mark
-74 wrUnderrun Write underrun occurred
-75 cantStepErr Drive error
-76 tk0BadErr Can't find track 0
-77 initIWMErr Can't initialize disk controller chip
-78 twoSideErr Tried to read side 2 of a disk in a single-sided drive
-79 spdAdjErr Can't correctly adjust disk speed
-80 seekErr Drive error
-81 sectNFErr Can't find sector

Also, to check for any low-level disk error:

-84 firstDskErr First of the range of low-level disk errors
-64 lastDskErr Last of the range of low-level disk errors

Clock Chip Errors

-85 clkRdErr Unable to read clock
-86 clkWrErr Time written did not verify
-87 prWrErr Parameter RAM written did not verify
-88 prInitErr Validity status is not $A8
AppleTalk Manager Errors

-91  ddpSktErr  DDP socket error: socket already active; not a well-known socket; socket table full; all dynamic socket numbers in use
-92  ddpLenErr  DDP datagram or ALAP data length too big
-93  noBridgeErr  No bridge found
-94  lapProtErr  ALAP error attaching/detaching ALAP protocol type: attach error when ALAP protocol type is negative, not in range, or already in table, or when table is full; detach error when ALAP protocol type isn't in table
-95  excessCollsns  ALAP no CTS received after 32 RTS's, or line sensed in use 32 times (not necessarily caused by collisions)
-97  portInUse  Driver Open error, port already in use
-98  portNotCf  Driver Open error, port not configured for this connection

Scrap Manager Errors

-100  noScrapiErr  Desk scrap isn't initialized
-102  noTypeErr  No data of the requested type

Memory Manager Errors

-108  memFullErr  Not enough room in heap zone
-109  iMemFullErr  Not enough room in heap zone (Printing Manager)
-109  nilHandleErr  NIL master pointer
-111  memWZErr  Attempt to operate on a free block
-112  memPurErr  Attempt to purge a locked block
-117  memLockedErr  Block is locked

Resource Manager Errors

-192  resNotFound  Resource not found
-193  resFNotFound  Resource file not found
-194  addResFailed  AddResource failed
-196  rmvResFailed  RmveResource failed

Additional AppleTalk Manager Errors

-1024  nbpBuffOvr  NBP buffer overflow
-1025  nbpNoConfirm  NBP name not confirmed
-1026  nbpConfDiff  NBP name confirmed for different socket
-1027  nbpDuplicate  NBP duplicate name already exists
-1028  nbpNotFound  NBP name not found
<table>
<thead>
<tr>
<th>Result Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1029</td>
<td>nbpNISErr</td>
</tr>
<tr>
<td></td>
<td>NBP names information socket error</td>
</tr>
<tr>
<td>-1096</td>
<td>reqFailed</td>
</tr>
<tr>
<td></td>
<td>ATPSndRequest failed: retry count exceeded</td>
</tr>
<tr>
<td>-1097</td>
<td>tooManyReqs</td>
</tr>
<tr>
<td></td>
<td>ATP too many concurrent requests</td>
</tr>
<tr>
<td>-1098</td>
<td>tooManySkts</td>
</tr>
<tr>
<td></td>
<td>ATP too many responding sockets</td>
</tr>
<tr>
<td>-1099</td>
<td>badATPSkt</td>
</tr>
<tr>
<td></td>
<td>ATP bad responding socket</td>
</tr>
<tr>
<td>-1100</td>
<td>badBuffNum</td>
</tr>
<tr>
<td></td>
<td>ATP bad sequence number</td>
</tr>
<tr>
<td>-1101</td>
<td>noRelErr</td>
</tr>
<tr>
<td></td>
<td>ATP no release received</td>
</tr>
<tr>
<td>-1102</td>
<td>cbNotFound</td>
</tr>
<tr>
<td></td>
<td>ATP control block not found</td>
</tr>
<tr>
<td>-1103</td>
<td>noSendResp</td>
</tr>
<tr>
<td></td>
<td>ATPAddRsp issued before ATPSndRsp</td>
</tr>
<tr>
<td>-1104</td>
<td>noDataArea</td>
</tr>
<tr>
<td></td>
<td>Too many outstanding ATP calls</td>
</tr>
<tr>
<td>-1105</td>
<td>reqAborted</td>
</tr>
<tr>
<td></td>
<td>Request aborted</td>
</tr>
<tr>
<td>-3101</td>
<td>buf2SmallErr</td>
</tr>
<tr>
<td></td>
<td>ALAP frame too large for buffer</td>
</tr>
<tr>
<td></td>
<td>DDP datagram too large for buffer</td>
</tr>
<tr>
<td>-3102</td>
<td>noMPPError</td>
</tr>
<tr>
<td></td>
<td>MPP driver not installed</td>
</tr>
<tr>
<td>-3103</td>
<td>cksumErr</td>
</tr>
<tr>
<td></td>
<td>DDP bad checksum</td>
</tr>
<tr>
<td>-3104</td>
<td>extractErr</td>
</tr>
<tr>
<td></td>
<td>NBP can't find tuple in buffer</td>
</tr>
<tr>
<td>-3105</td>
<td>readQErr</td>
</tr>
<tr>
<td></td>
<td>Socket or protocol type invalid or not found in table</td>
</tr>
<tr>
<td>-3106</td>
<td>atpLenErr</td>
</tr>
<tr>
<td></td>
<td>ATP response message too large</td>
</tr>
<tr>
<td>-3107</td>
<td>atpBadRsp</td>
</tr>
<tr>
<td></td>
<td>Bad response from ATPRequest</td>
</tr>
<tr>
<td>-3108</td>
<td>recNotFnd</td>
</tr>
<tr>
<td></td>
<td>ABRecord not found</td>
</tr>
<tr>
<td>-3109</td>
<td>sktClosedErr</td>
</tr>
<tr>
<td></td>
<td>Asynchronous call aborted because socket was closed before call was completed</td>
</tr>
</tbody>
</table>
APPENDIX B: ROUTINES THAT MAY MOVE OR PURGE MEMORY

This appendix lists all the routines that may move or purge blocks in the heap. As described in chapter 1 of Volume II, calling these routines may cause problems if a handle has been dereferenced. None of these routines may be called from within an interrupt, such as in a completion routine or a VBL task.

The Pascal name of each routine is shown, except for a few cases where there's no Pascal interface corresponding to a particular trap; in those cases, the trap macro name is shown instead (without its initial underscore character).

AddResMenu  CopyBits  DrawPicture
Alert       CopyRgn     DrawString
AppendMenu  CouldAlert  DrawText
ATPAddRsp   CouldDialog  DriveStatus
ATPCloseSocket  CreateResFile  DrvRemove
ATPGetRequest  DDPCloseSocket  EaseArc
ATPLoad     DDPOpenSocket  EraseArc
ATPOpenSocket  DDPRdCancel  EraseOval
ATPRRequest  DDPRead     ErasePoly
ATPRspCancel  DDPRWrite   EraseRect
ATFSndRequest  DialogSelect  EraseRgn
ATPSndRsp    DiffRgn     EraseRoundRect
ATPUntload  DiffFormat   EventAvail
BeginUpdate  DIFormat    ExitToShell
BringToFront  DILoad      FillArc
Button       DiskEject   FillIval
CalcMenuSize  DisposeDialog  FillPoly
CalcVis      DisposeControl  FillRect
CalcVisBehind  DisposeMenu  FillRgn
CautionAlert  DisposeRgn    FillRoundRect
Chain        DisposeWindow  FindControl
ChangedResource  DisposeHandle  FlashMenuBar
CharWidth    DisposePtr     FlushVol
CheckItem    DIUnload     FMSwapFont
CheckUpdate  DVERIFY      FrameArc
ClipAbove    DigCopy       FrameOval
ClipRect     DigCut        FramePoly
CloseDialog  DigDelete    FrameRect
ClosePicture  DigPaste    FrameRgn
ClosePoly    DragControl  FrameRoundRect
ClosePort    DragGrayRgn  FreeAlert
CloseResFile  DragWindow   FreeDialog
CloseRgn     DrawChar     FreeMem
CloseWindow  DrawDialog   GetClip
CompactMem  DrawGrowIcon  GetCursor
Control  DrawMenuBar

Routines That May Move or Purge Memory III-211
GetDCtrlEntry
GetDItem
GetFNum
GetFontInfo
GetFontName
GetIcon
GetIndPattern
GetIndResource
GetIndString
GetKeys
GetMenu
GetMenuBar
GetNamedResource
GetNewControl
GetNewDialog
GetNewMBar
GetNewWindow
GetNextEvent
GetPattern
GetPicture
GetResource
GetScrap
GetString
GrowWindow
HandAndHand
HandToHand
HideControl
HideWindow
HiliteControl
HiliteMenu
HiliteWindow
InitAllPacks
InitApplZone
InitFonts
InitMenus
InitPack
InitPort
InitResources
InitWindows
InitZone
InsertMenu
InsertResMenu
InsertRgn
InvalRect
InvalRgn
InvertArc
InvertOval
InvertPoly
InvertRect
InvertRgn
InvertRoundRect
IUCompString
IUDatePString
IUDateString
IUEqualString
IUGetIntl
IUMagIDString
IUMagString
IUMetric
IUSetIntl
IUTimePString
IUTimeString
KillControls
KillPicture
KillPoly
LAPCloseProtocol
LAPOpenProtocol
LAPrdCancel
LAPRead
LAPWrite
Launch
Line
LineTo
LoadResource
LoadScrap
LoadSeg
MapRgn
MenuKey
MenuSelect
ModalDialog
MoreMasters
MoveControl
MoveHHI
MoveWindow
MPPClose
MPPOpen
Munger
NBPConfirm
NPBExtract
NBPLoad
NBPLookup
NBPRegister
NBPRemove
NBPUnload
NewControl
NewDialog
NewHandle
NewMenu
NewPtr
NewRgn
NewString
NewWindow
NoteAlert
NumToString
OpenDeskAcc
OpenPicture
OpenPoly
OpenPort
OpenResFile
OpenRgn
PaintArc
PaintBehind
PaintOne
PaintOval
PaintPoly
PaintRect
PaintRgn
PaintRoundRect
ParamText
PBControl
PBEject
PBFlushVol
PBMountVol
PBOffLine
PBOpen
PBOpenRF
PBStatus
PicComment
PlotIcon
PrClose
PrCloseDoc
PrClosePage
PrCtlCall
PrDrvrDCE
PrDrvrVers
PrintDefault
PrJobDialog
PrJobMerge
PrOpen
PrOpenDoc
PrOpenPage
PrPicFile
PrStdDialog
PrValidate
PtrAndHand
PtrToHand
PtrToXHand
PurgeMem
PutScrap
RAMSDClose
RAMSDOpen
RealFont
ReallocHandle
RecoverHandle
RectRgn

III-212 Routines That May Move or Purge Memory
### Routines That May Move or Purge Memory

<table>
<thead>
<tr>
<th>ReleaseResource</th>
<th>SetString</th>
<th>TEActivate</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResrvMem</td>
<td>SetTagBuffer</td>
<td>TECalText</td>
</tr>
<tr>
<td>Restart</td>
<td>SetWTitle</td>
<td>TEClick</td>
</tr>
<tr>
<td>RmveResource</td>
<td>SFGetFile</td>
<td>TECopy</td>
</tr>
<tr>
<td>RsrcZoneInit</td>
<td>SFGetFile</td>
<td>TECut</td>
</tr>
<tr>
<td>SaveOld</td>
<td>SFPutFile</td>
<td>TEDeactivate</td>
</tr>
<tr>
<td>ScrollRect</td>
<td>ShowControl</td>
<td>TEDelete</td>
</tr>
<tr>
<td>SectRgn</td>
<td>ShowHide</td>
<td>TEDispose</td>
</tr>
<tr>
<td>SelectWindow</td>
<td>ShowWindow</td>
<td>TEFFromScrap</td>
</tr>
<tr>
<td>SelIText</td>
<td>SizeControl</td>
<td>TEGGetText</td>
</tr>
<tr>
<td>SendBehind</td>
<td>SizeWindow</td>
<td>TELIdle</td>
</tr>
<tr>
<td>SerClrBrk</td>
<td>StartSound</td>
<td>TELInit</td>
</tr>
<tr>
<td>SerGetBrk</td>
<td>Status</td>
<td>TELInsert</td>
</tr>
<tr>
<td>SerHShake</td>
<td>StdArc</td>
<td>TESetJust</td>
</tr>
<tr>
<td>SerReset</td>
<td>StdBits</td>
<td>TESetSelect</td>
</tr>
<tr>
<td>SerSetBrk</td>
<td>StdComment</td>
<td>TESetText</td>
</tr>
<tr>
<td>SerSetBuf</td>
<td>StdLine</td>
<td>TestControl</td>
</tr>
<tr>
<td>SerStatus</td>
<td>StdOval</td>
<td>TEScroll</td>
</tr>
<tr>
<td>SetApplBase</td>
<td>StdPoly</td>
<td>TESetSelect</td>
</tr>
<tr>
<td>SetClip</td>
<td>StdPutPic</td>
<td>TESetSelect</td>
</tr>
<tr>
<td>SetCTitle</td>
<td>StdRect</td>
<td>TESetText</td>
</tr>
<tr>
<td>SetCtlMax</td>
<td>StdRgn</td>
<td>TestControl</td>
</tr>
<tr>
<td>SetCtlMin</td>
<td>StdRRect</td>
<td>TEToScrap</td>
</tr>
<tr>
<td>SetCtlValue</td>
<td>StdText</td>
<td>TEUpdate</td>
</tr>
<tr>
<td>SetDItem</td>
<td>StdTxMeas</td>
<td>TextBox</td>
</tr>
<tr>
<td>SetEmptyRgn</td>
<td>StillDown</td>
<td>TextWidth</td>
</tr>
<tr>
<td>SetFontLock</td>
<td>StopAlert</td>
<td>TickCount</td>
</tr>
<tr>
<td>SetHandleSize</td>
<td>StopSound</td>
<td>TrackControl</td>
</tr>
<tr>
<td>SetItem</td>
<td>StringToNum</td>
<td>TrackGoAway</td>
</tr>
<tr>
<td>SetItemIcon</td>
<td>StringComparison</td>
<td>UnionRgn</td>
</tr>
<tr>
<td>SetItemMark</td>
<td>StringWidth</td>
<td>UploadScrap</td>
</tr>
<tr>
<td>SetItemStyle</td>
<td>SysBeep</td>
<td>UploadSeg</td>
</tr>
<tr>
<td>SetText</td>
<td>SysError</td>
<td>ValidRect</td>
</tr>
<tr>
<td>SetPtrSize</td>
<td>SystemClick</td>
<td>ValidRgn</td>
</tr>
<tr>
<td>SetRectRgn</td>
<td>SystemEdit</td>
<td>WaitForMouseUp</td>
</tr>
<tr>
<td>SetResInfo</td>
<td>SystemMenu</td>
<td>XorRgn</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ZeroScrap</td>
</tr>
</tbody>
</table>
Inside Macintosh
APPENDIX C: SYSTEM TRAPS

This appendix lists the trap macros for the Toolbox and Operating System routines and their corresponding trap word values in hexadecimal. The "Name" column gives the trap macro name (without its initial underscore character). In those cases where the name of the equivalent Pascal call is different, the Pascal name appears indented under the main entry. The routines in Macintosh packages are listed under the macros they invoke after pushing a routine selector onto the stack; the routine selector follows the Pascal routine name in parentheses.

There are two tables: The first is ordered alphabetically by name; the second is ordered numerically by trap number, for use when debugging. (The trap number is the last two digits of the trap word unless the trap word begins with A9, in which case the trap number is 1 followed by the last two digits of the trap word.)

Note: The Operating System Utility routines GetTrapAddress and SetTrapAddress take a trap number as a parameter, not a trap word.

Warning: Traps that aren't currently used by the system are reserved for future use.

<table>
<thead>
<tr>
<th>Name</th>
<th>Trap word</th>
<th>Name</th>
<th>Trap word</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddDrive</td>
<td>A04E</td>
<td>ChangedResource</td>
<td>A9AA</td>
</tr>
<tr>
<td>(internal use only)</td>
<td></td>
<td>CharWidth</td>
<td>A88D</td>
</tr>
<tr>
<td>AddPt</td>
<td>A87E</td>
<td>CheckItem</td>
<td>A945</td>
</tr>
<tr>
<td>AddResMenu</td>
<td>A94D</td>
<td>CheckUpdate</td>
<td>A911</td>
</tr>
<tr>
<td>AddResource</td>
<td>A9AB</td>
<td>ClearMenuBar</td>
<td>A934</td>
</tr>
<tr>
<td>Alert</td>
<td>A985</td>
<td>ClipAbove</td>
<td>A90B</td>
</tr>
<tr>
<td>Allocate</td>
<td>A010</td>
<td>ClipRect</td>
<td>A87B</td>
</tr>
<tr>
<td>PBAllocate</td>
<td></td>
<td>Close</td>
<td>A001</td>
</tr>
<tr>
<td>AngleFromSlope</td>
<td>A8C4</td>
<td>PBClose</td>
<td>A9B7</td>
</tr>
<tr>
<td>AppendMenu</td>
<td>A933</td>
<td>CloseDeskAcc</td>
<td>A9B7</td>
</tr>
<tr>
<td>BackColor</td>
<td>A863</td>
<td>CloseDialog</td>
<td>A982</td>
</tr>
<tr>
<td>BackPat</td>
<td>A87C</td>
<td>ClosePgon</td>
<td>A8CC</td>
</tr>
<tr>
<td>BeginUpdate</td>
<td>A922</td>
<td>ClosePoly</td>
<td>A8CC</td>
</tr>
<tr>
<td>BitAnd</td>
<td>A858</td>
<td>ClosePicture</td>
<td>A8F4</td>
</tr>
<tr>
<td>BitClr</td>
<td>A85F</td>
<td>ClosePort</td>
<td>A87D</td>
</tr>
<tr>
<td>BitNot</td>
<td>A85A</td>
<td>CloseResFile</td>
<td>A99A</td>
</tr>
<tr>
<td>BitOr</td>
<td>A85B</td>
<td>CloseRgn</td>
<td>A8DB</td>
</tr>
<tr>
<td>BitSet</td>
<td>A85E</td>
<td>CloseWindow</td>
<td>A92D</td>
</tr>
<tr>
<td>BitShift</td>
<td>A85C</td>
<td>CmpString</td>
<td>A03C</td>
</tr>
<tr>
<td>BitTst</td>
<td>A85D</td>
<td>EqualString</td>
<td>A864</td>
</tr>
<tr>
<td>BitXor</td>
<td>A859</td>
<td>ColorBit</td>
<td>A864</td>
</tr>
<tr>
<td>BlockMove</td>
<td>A02E</td>
<td>CompactMem</td>
<td>A04C</td>
</tr>
<tr>
<td>BringToFront</td>
<td>A920</td>
<td>Control</td>
<td>A004</td>
</tr>
<tr>
<td>Button</td>
<td>A974</td>
<td>PBControl</td>
<td>A8EC</td>
</tr>
<tr>
<td>CalcMenuSize</td>
<td>A948</td>
<td>CopyBits</td>
<td>A8EC</td>
</tr>
<tr>
<td>CalcVBehind</td>
<td>A90A</td>
<td>CopyRgn</td>
<td>A8DC</td>
</tr>
<tr>
<td>CalcVis</td>
<td>A909</td>
<td>CouldAlert</td>
<td>A989</td>
</tr>
<tr>
<td>CautionAlert</td>
<td>A988</td>
<td>CouldDialog</td>
<td>A979</td>
</tr>
<tr>
<td>Chain</td>
<td>A9F3</td>
<td>CountMItems</td>
<td>A950</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CountResources</td>
<td>A99C</td>
</tr>
<tr>
<td>Name</td>
<td>Trap word</td>
<td>Name</td>
<td>Trap word</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------</td>
<td>-----------------------</td>
<td>-----------</td>
</tr>
<tr>
<td>CountTypes</td>
<td>A99E</td>
<td>EndUpdate</td>
<td>A923</td>
</tr>
<tr>
<td>Create</td>
<td>A008</td>
<td>Enqueue</td>
<td>A96F</td>
</tr>
<tr>
<td>PBCreate</td>
<td>A9B1</td>
<td>EqualPt</td>
<td>A881</td>
</tr>
<tr>
<td>CreateResFile</td>
<td>A994</td>
<td>EqualRect</td>
<td>A8A6</td>
</tr>
<tr>
<td>CurResFile</td>
<td>A9C7</td>
<td>EqualRgn</td>
<td>A8E3</td>
</tr>
<tr>
<td>Date2Secs</td>
<td>A823</td>
<td>EraseArc</td>
<td>A8C0</td>
</tr>
<tr>
<td>Delay</td>
<td>A03B</td>
<td>EraseEoval</td>
<td>A8B9</td>
</tr>
<tr>
<td>Delete</td>
<td>A009</td>
<td>ErasePoly</td>
<td>A8C8</td>
</tr>
<tr>
<td>PBDelete</td>
<td>A96F</td>
<td>EraseRect</td>
<td>A8A3</td>
</tr>
<tr>
<td>DeleteMenu</td>
<td>A936</td>
<td>EraseRgn</td>
<td>A8D4</td>
</tr>
<tr>
<td>DeltaPoint</td>
<td>A94F</td>
<td>EraseRoundRect</td>
<td>A8B2</td>
</tr>
<tr>
<td>Dequeue</td>
<td>A96E</td>
<td>ErrorSound</td>
<td>A98C</td>
</tr>
<tr>
<td>DetachResource</td>
<td>A992</td>
<td>EventAvail</td>
<td>A971</td>
</tr>
<tr>
<td>DialogSelect</td>
<td>A980</td>
<td>ExitToShell</td>
<td>A9F4</td>
</tr>
<tr>
<td>DiffRgn</td>
<td>A8E6</td>
<td>FillArc</td>
<td>A8C2</td>
</tr>
<tr>
<td>DisableItem</td>
<td>A93A</td>
<td>FillOval</td>
<td>A8BB</td>
</tr>
<tr>
<td>DisposeControl</td>
<td>A955</td>
<td>FillPoly</td>
<td>A8CA</td>
</tr>
<tr>
<td>DisposeControl</td>
<td>A983</td>
<td>FillRect</td>
<td>A8A5</td>
</tr>
<tr>
<td>DisposeDialog</td>
<td>A023</td>
<td>FillRgn</td>
<td>A8D6</td>
</tr>
<tr>
<td>DisposeHandle</td>
<td>A932</td>
<td>FillRoundRect</td>
<td>A8B4</td>
</tr>
<tr>
<td>DisposeMenu</td>
<td>A01F</td>
<td>FindControl</td>
<td>A96C</td>
</tr>
<tr>
<td>DisposeMenu</td>
<td>A8D9</td>
<td>FindWindow</td>
<td>A92C</td>
</tr>
<tr>
<td>DisposePtr</td>
<td>A914</td>
<td>FixMul</td>
<td>A868</td>
</tr>
<tr>
<td>DisposeRgn</td>
<td>DispostWindow</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DisposeWindow</td>
<td>DispostWindow</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DragControl</td>
<td>A967</td>
<td>FixRatio</td>
<td>A869</td>
</tr>
<tr>
<td>DragGrayRgn</td>
<td>A905</td>
<td>FixRound</td>
<td>A86C</td>
</tr>
<tr>
<td>DragTheRgn</td>
<td>A926</td>
<td>FlashMenuBar</td>
<td>A94C</td>
</tr>
<tr>
<td>DragWindow</td>
<td>A925</td>
<td>FlushEvents</td>
<td>A032</td>
</tr>
<tr>
<td>DrawChar</td>
<td>A883</td>
<td>FlushFile</td>
<td>A045</td>
</tr>
<tr>
<td>DrawControls</td>
<td>A969</td>
<td>PBFlushFile</td>
<td>A013</td>
</tr>
<tr>
<td>DrawDialog</td>
<td>A981</td>
<td>PBFlushVol</td>
<td></td>
</tr>
<tr>
<td>DrawGrowlcon</td>
<td>A904</td>
<td>FMSwapFont</td>
<td>A901</td>
</tr>
<tr>
<td>DrawMenuBar</td>
<td>A937</td>
<td>ForeColor</td>
<td>A862</td>
</tr>
<tr>
<td>DrawNew</td>
<td>A90F</td>
<td>FP68K</td>
<td>A9EB</td>
</tr>
<tr>
<td>DrawPicture</td>
<td>A8F6</td>
<td>FrameArc</td>
<td>A8BE</td>
</tr>
<tr>
<td>DrawString</td>
<td>A884</td>
<td>FrameOval</td>
<td>A8B7</td>
</tr>
<tr>
<td>DrawText</td>
<td>A885</td>
<td>FramePoly</td>
<td>A8C6</td>
</tr>
<tr>
<td>DrvrInstall</td>
<td>A03D</td>
<td>FrameRect</td>
<td>A8A1</td>
</tr>
<tr>
<td>(internal use only)</td>
<td></td>
<td>FrameRgn</td>
<td>A8D2</td>
</tr>
<tr>
<td>DrvrRemove</td>
<td>A03E</td>
<td>FrameRoundRect</td>
<td>A8B0</td>
</tr>
<tr>
<td>(internal use only)</td>
<td></td>
<td>FreeAlert</td>
<td>A98A</td>
</tr>
<tr>
<td>Eject</td>
<td>A017</td>
<td>FreeDialog</td>
<td>A97A</td>
</tr>
<tr>
<td>PBDoor</td>
<td></td>
<td>FreeMem</td>
<td>A01C</td>
</tr>
<tr>
<td>EmptyRgn</td>
<td>A9EC</td>
<td>FrontWindow</td>
<td>A924</td>
</tr>
<tr>
<td>EmptyHandle</td>
<td>A02B</td>
<td>GetAppParms</td>
<td>A9F5</td>
</tr>
<tr>
<td>EmptyRect</td>
<td>A8AE</td>
<td>GetClip</td>
<td>A87A</td>
</tr>
<tr>
<td>EmptyRgn</td>
<td>A8E2</td>
<td>GetDrawRefCon</td>
<td>A95A</td>
</tr>
<tr>
<td>EnableItem</td>
<td>A939</td>
<td>GetCTitle</td>
<td>A95E</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GetCtAction</td>
<td>A96A</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GetCtlValue</td>
<td>A960</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GetCursor</td>
<td>A9B9</td>
</tr>
<tr>
<td>Name</td>
<td>Trap word</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetDlgItem</td>
<td>A98D</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetE0F</td>
<td>A011</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PBGetEOF</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetFilelnfo</td>
<td>A00C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PBGetFilelnfo</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetFName</td>
<td>A8FF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetFontName</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetFNum</td>
<td>A900</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetFontInfo</td>
<td>A88B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetFPos</td>
<td>A018</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PBGetFPos</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetHandleSize</td>
<td>A025</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetIcon</td>
<td>A9BB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetIndlnfo</td>
<td>A99D</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetIndType</td>
<td>A99F</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetItem</td>
<td>A946</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetIText</td>
<td>A990</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetITmIcon</td>
<td>A93F</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetITmMark</td>
<td>A943</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetITmStyle</td>
<td>A941</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetKeys</td>
<td>A976</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetMaxCtl</td>
<td>A962</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetCtlMax</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetMenuBar</td>
<td>A93B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetMHandle</td>
<td>A949</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetMinCtl</td>
<td>A961</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetCtlMin</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetMouse</td>
<td>A972</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetNamedResource</td>
<td>A9A1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetNewControl</td>
<td>A9BE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetNewDialog</td>
<td>A97C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetNewMBar</td>
<td>A9C0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetNewWindow</td>
<td>A9BD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetNextEvent</td>
<td>A970</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetOSEvent</td>
<td>A031</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetPattern</td>
<td>A9B8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetPen</td>
<td>A89A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetPenState</td>
<td>A898</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetPicture</td>
<td>A9BC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetPixel</td>
<td>A865</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetPort</td>
<td>A874</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetPtrSize</td>
<td>A021</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetResAttrs</td>
<td>A9A6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetResFileAttrs</td>
<td>A9F6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetReslnfo</td>
<td>A9A8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetResource</td>
<td>A9A0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetRMenu</td>
<td>A9BF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetMenu</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Trap word</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetScrap</td>
<td>A9FD</td>
</tr>
<tr>
<td>GetString</td>
<td>A9BA</td>
</tr>
<tr>
<td>GetTrapAddress</td>
<td>A146</td>
</tr>
<tr>
<td>GetVol</td>
<td>A014</td>
</tr>
<tr>
<td>PBGetVol</td>
<td></td>
</tr>
<tr>
<td>GetVolInfo</td>
<td>A007</td>
</tr>
<tr>
<td>PBGetVolInfo</td>
<td></td>
</tr>
<tr>
<td>GetWindowPic</td>
<td>A92F</td>
</tr>
<tr>
<td>GetWMgrPort</td>
<td>A910</td>
</tr>
<tr>
<td>GetWRefCon</td>
<td>A917</td>
</tr>
<tr>
<td>GetWTitle</td>
<td>A919</td>
</tr>
<tr>
<td>GetZone</td>
<td>A11A</td>
</tr>
<tr>
<td>GlobalToLocal</td>
<td>A871</td>
</tr>
<tr>
<td>GrafDevice</td>
<td>A872</td>
</tr>
<tr>
<td>GrowWindow</td>
<td>A92B</td>
</tr>
<tr>
<td>HandAndHand</td>
<td>A9E4</td>
</tr>
<tr>
<td>HandleZone</td>
<td>A126</td>
</tr>
<tr>
<td>HandToHand</td>
<td>A9E1</td>
</tr>
<tr>
<td>HideControl</td>
<td>A958</td>
</tr>
<tr>
<td>HideCursor</td>
<td>A852</td>
</tr>
<tr>
<td>HidePen</td>
<td>A896</td>
</tr>
<tr>
<td>HideWindow</td>
<td>A916</td>
</tr>
<tr>
<td>HiliteControl</td>
<td>A95D</td>
</tr>
<tr>
<td>HiliteMenu</td>
<td>A938</td>
</tr>
<tr>
<td>HiliteWindow</td>
<td>A91C</td>
</tr>
<tr>
<td>HiWord</td>
<td>A86A</td>
</tr>
<tr>
<td>HLock</td>
<td>A029</td>
</tr>
<tr>
<td>HNoPurge</td>
<td>A04A</td>
</tr>
<tr>
<td>HomeResFile</td>
<td>A9A4</td>
</tr>
<tr>
<td>HPurge</td>
<td>A049</td>
</tr>
<tr>
<td>HUnlock</td>
<td>A02A</td>
</tr>
<tr>
<td>InfoScrap</td>
<td>A9F9</td>
</tr>
<tr>
<td>InitAllPacks</td>
<td>A9E6</td>
</tr>
<tr>
<td>InitApplZone</td>
<td>A02C</td>
</tr>
<tr>
<td>InitCursor</td>
<td>A850</td>
</tr>
<tr>
<td>InitDialogs</td>
<td>A97B</td>
</tr>
<tr>
<td>InitFonts</td>
<td>A8FE</td>
</tr>
<tr>
<td>InitGraf</td>
<td>A86E</td>
</tr>
<tr>
<td>InitMenus</td>
<td>A930</td>
</tr>
<tr>
<td>InitPack</td>
<td>A9E5</td>
</tr>
<tr>
<td>InitPort</td>
<td>A86D</td>
</tr>
<tr>
<td>InitQueue</td>
<td>A016</td>
</tr>
<tr>
<td>FInitQueue</td>
<td></td>
</tr>
<tr>
<td>InitResources</td>
<td>A995</td>
</tr>
<tr>
<td>InitUtil</td>
<td>A03F</td>
</tr>
<tr>
<td>InitWindows</td>
<td>A912</td>
</tr>
<tr>
<td>InitZone</td>
<td>A019</td>
</tr>
<tr>
<td>InsertMenu</td>
<td>A935</td>
</tr>
<tr>
<td>InsertResMenu</td>
<td>A951</td>
</tr>
<tr>
<td>InsetRect</td>
<td>A8A9</td>
</tr>
<tr>
<td>InsetRgn</td>
<td>A8E1</td>
</tr>
</tbody>
</table>

System Traps III-217
<table>
<thead>
<tr>
<th>Name</th>
<th>Trap word</th>
</tr>
</thead>
<tbody>
<tr>
<td>InvalRect</td>
<td>A928</td>
</tr>
<tr>
<td>InvalRgn</td>
<td>A927</td>
</tr>
<tr>
<td>InverRect</td>
<td>A8A4</td>
</tr>
<tr>
<td>InvertRect</td>
<td>A8D5</td>
</tr>
<tr>
<td>InverRgn</td>
<td>A87A</td>
</tr>
<tr>
<td>InverRoundRect</td>
<td>A8B3</td>
</tr>
<tr>
<td>InvertArc</td>
<td>A8C1</td>
</tr>
<tr>
<td>InvertOval</td>
<td>A8BA</td>
</tr>
<tr>
<td>InvertPoly</td>
<td>A8C9</td>
</tr>
<tr>
<td>IsDialogEvent</td>
<td>A97F</td>
</tr>
<tr>
<td>KillControls</td>
<td>A956</td>
</tr>
<tr>
<td>KillIO</td>
<td>A006</td>
</tr>
<tr>
<td>PBKillIO</td>
<td>A8F5</td>
</tr>
<tr>
<td>KillPicture</td>
<td>A8CD</td>
</tr>
<tr>
<td>KillPoly</td>
<td>A9F2</td>
</tr>
<tr>
<td>Launch</td>
<td>A991</td>
</tr>
<tr>
<td>Line</td>
<td>A892</td>
</tr>
<tr>
<td>LineTo</td>
<td>A8A2</td>
</tr>
<tr>
<td>LoadResource</td>
<td>A9A2</td>
</tr>
<tr>
<td>LoadSeg</td>
<td>A9F0</td>
</tr>
<tr>
<td>LocalToGlobal</td>
<td>A870</td>
</tr>
<tr>
<td>LodeScrap</td>
<td>A9FB</td>
</tr>
<tr>
<td>LoadScrap</td>
<td>A8A7</td>
</tr>
<tr>
<td>LongMul</td>
<td>A867</td>
</tr>
<tr>
<td>LoWord</td>
<td>A86B</td>
</tr>
<tr>
<td>MapPoly</td>
<td>A8FC</td>
</tr>
<tr>
<td>MapPt</td>
<td>A8F9</td>
</tr>
<tr>
<td>MapRect</td>
<td>A8FA</td>
</tr>
<tr>
<td>MapRgn</td>
<td>A8FB</td>
</tr>
<tr>
<td>MaxMem</td>
<td>A11D</td>
</tr>
<tr>
<td>MenuKey</td>
<td>A93E</td>
</tr>
<tr>
<td>MenuSelect</td>
<td>A93D</td>
</tr>
<tr>
<td>ModalDialog</td>
<td>A991</td>
</tr>
<tr>
<td>MoreMasters</td>
<td>A036</td>
</tr>
<tr>
<td>MountVol</td>
<td>A00F</td>
</tr>
<tr>
<td>PBMountVol</td>
<td>A894</td>
</tr>
<tr>
<td>Move</td>
<td>A959</td>
</tr>
<tr>
<td>MoveControl</td>
<td>A877</td>
</tr>
<tr>
<td>MovePortTo</td>
<td>A893</td>
</tr>
<tr>
<td>MoveTo</td>
<td>A91B</td>
</tr>
<tr>
<td>MoveWindow</td>
<td>A9E0</td>
</tr>
<tr>
<td>Munger</td>
<td>A954</td>
</tr>
<tr>
<td>NewControl</td>
<td>A97D</td>
</tr>
<tr>
<td>NewDialog</td>
<td>A122</td>
</tr>
<tr>
<td>NewHandle</td>
<td>A931</td>
</tr>
<tr>
<td>NewMenu</td>
<td>A11E</td>
</tr>
<tr>
<td>NewPtr</td>
<td>A8D8</td>
</tr>
<tr>
<td>NewRgn</td>
<td>A906</td>
</tr>
<tr>
<td>NewString</td>
<td>A913</td>
</tr>
<tr>
<td>NoteAlert</td>
<td>A987</td>
</tr>
<tr>
<td>ObscurCursor</td>
<td>A856</td>
</tr>
<tr>
<td>Offline</td>
<td>A035</td>
</tr>
<tr>
<td>PBOffline</td>
<td>A8CE</td>
</tr>
<tr>
<td>OffsetRect</td>
<td>A88A</td>
</tr>
<tr>
<td>OffsetRgn</td>
<td>A8E0</td>
</tr>
<tr>
<td>Open</td>
<td>A000</td>
</tr>
<tr>
<td>PBOpen</td>
<td>A9B6</td>
</tr>
<tr>
<td>OpenDeskAcc</td>
<td>A8F3</td>
</tr>
<tr>
<td>OpenPicture</td>
<td>A8CB</td>
</tr>
<tr>
<td>OpenPoly</td>
<td>A86F</td>
</tr>
<tr>
<td>OpenPort</td>
<td>A997</td>
</tr>
<tr>
<td>OpenResFile</td>
<td>A00A</td>
</tr>
<tr>
<td>OpenRF</td>
<td>A8DA</td>
</tr>
<tr>
<td>OSEventAvail</td>
<td>A030</td>
</tr>
<tr>
<td>Pack0</td>
<td>A9E7</td>
</tr>
<tr>
<td>(reserved for future use)</td>
<td></td>
</tr>
<tr>
<td>Pack1</td>
<td>A9E8</td>
</tr>
<tr>
<td>(reserved for future use)</td>
<td></td>
</tr>
<tr>
<td>Pack2</td>
<td>A9E9</td>
</tr>
<tr>
<td>Pack3</td>
<td>A9EA</td>
</tr>
<tr>
<td>Pack4</td>
<td>A9EB</td>
</tr>
<tr>
<td>Pack5</td>
<td>A9EC</td>
</tr>
<tr>
<td>Pack6</td>
<td>A9ED</td>
</tr>
<tr>
<td>Pack7</td>
<td>A9EE</td>
</tr>
<tr>
<td>NumToString</td>
<td>A900</td>
</tr>
<tr>
<td>StringToNum</td>
<td>A901</td>
</tr>
</tbody>
</table>

**III-218 System Traps**
<table>
<thead>
<tr>
<th>Name</th>
<th>Trap word</th>
</tr>
</thead>
<tbody>
<tr>
<td>PackBits</td>
<td>A8CF</td>
</tr>
<tr>
<td>PaintArc</td>
<td>A8BF</td>
</tr>
<tr>
<td>PaintBehind</td>
<td>A90D</td>
</tr>
<tr>
<td>PaintOne</td>
<td>A90C</td>
</tr>
<tr>
<td>PaintOval</td>
<td>A8B8</td>
</tr>
<tr>
<td>PaintPoly</td>
<td>A8C7</td>
</tr>
<tr>
<td>PaintRect</td>
<td>A8A2</td>
</tr>
<tr>
<td>PaintRgn</td>
<td>A8D3</td>
</tr>
<tr>
<td>PaintRoundRect</td>
<td>A8B1</td>
</tr>
<tr>
<td>ParamText</td>
<td>A98B</td>
</tr>
<tr>
<td>PenMode</td>
<td>A89C</td>
</tr>
<tr>
<td>PenNormal</td>
<td>A89E</td>
</tr>
<tr>
<td>PenPat</td>
<td>A89D</td>
</tr>
<tr>
<td>PenSize</td>
<td>A89B</td>
</tr>
<tr>
<td>PicComment</td>
<td>A8F2</td>
</tr>
<tr>
<td>PinRect</td>
<td>A94E</td>
</tr>
<tr>
<td>Plotlcon</td>
<td>A94B</td>
</tr>
<tr>
<td>PortSize</td>
<td>A876</td>
</tr>
<tr>
<td>PostEvent</td>
<td>A02F</td>
</tr>
<tr>
<td>Pr2Rect</td>
<td>A8AC</td>
</tr>
<tr>
<td>PrInRect</td>
<td>A8AD</td>
</tr>
<tr>
<td>PrInRgn</td>
<td>A8E8</td>
</tr>
<tr>
<td>PtrAndHand</td>
<td>A9E6</td>
</tr>
<tr>
<td>PtrToHand</td>
<td>A9E3</td>
</tr>
<tr>
<td>PtrToXHand</td>
<td>A9E2</td>
</tr>
<tr>
<td>PtrZone</td>
<td>A148</td>
</tr>
<tr>
<td>PtrToAngle</td>
<td>A8C3</td>
</tr>
<tr>
<td>PurgeMem</td>
<td>A04D</td>
</tr>
<tr>
<td>PutScrap</td>
<td>A9FE</td>
</tr>
<tr>
<td>Random</td>
<td>A861</td>
</tr>
<tr>
<td>RDrvrlnstall</td>
<td>A04F</td>
</tr>
<tr>
<td>Read</td>
<td>A002</td>
</tr>
<tr>
<td>PBRead</td>
<td>A039</td>
</tr>
<tr>
<td>ReadDateTime</td>
<td>A092</td>
</tr>
<tr>
<td>RealFont</td>
<td>A027</td>
</tr>
<tr>
<td>ReallocHandle</td>
<td>A128</td>
</tr>
<tr>
<td>RecoverHandle</td>
<td>A8E9</td>
</tr>
<tr>
<td>RectInRgn</td>
<td>A8DF</td>
</tr>
<tr>
<td>RectRgn</td>
<td>A9A3</td>
</tr>
<tr>
<td>ReleaseResource</td>
<td>A00B</td>
</tr>
<tr>
<td>Rename</td>
<td>A9AF</td>
</tr>
<tr>
<td>ReslError</td>
<td>A040</td>
</tr>
<tr>
<td>ResrvMem</td>
<td>A9AD</td>
</tr>
<tr>
<td>RmveResource</td>
<td>A996</td>
</tr>
<tr>
<td>RsrcZonelnit</td>
<td>A042</td>
</tr>
<tr>
<td>RstFilLock</td>
<td>A90E</td>
</tr>
<tr>
<td>ScalePt</td>
<td>A8F8</td>
</tr>
<tr>
<td>Name</td>
<td>Trap word</td>
</tr>
<tr>
<td>ScrollRect</td>
<td>A8EF</td>
</tr>
<tr>
<td>Secs2Date</td>
<td>A9C6</td>
</tr>
<tr>
<td>SectRect</td>
<td>A8AA</td>
</tr>
<tr>
<td>SectRgn</td>
<td>A8E4</td>
</tr>
<tr>
<td>SelectWindow</td>
<td>A91F</td>
</tr>
<tr>
<td>SetIText</td>
<td>A97E</td>
</tr>
<tr>
<td>SendBehind</td>
<td>A921</td>
</tr>
<tr>
<td>SetAppBase</td>
<td>A057</td>
</tr>
<tr>
<td>(internal use only)</td>
<td></td>
</tr>
<tr>
<td>SetAppBase</td>
<td>A02D</td>
</tr>
<tr>
<td>SetClip</td>
<td>A879</td>
</tr>
<tr>
<td>SetCREfCon</td>
<td>A95B</td>
</tr>
<tr>
<td>SetCTitle</td>
<td>A95F</td>
</tr>
<tr>
<td>SetCtlAction</td>
<td>A96B</td>
</tr>
<tr>
<td>SetCtlValue</td>
<td>A963</td>
</tr>
<tr>
<td>SetCursor</td>
<td>A851</td>
</tr>
<tr>
<td>SetDateTime</td>
<td>A03A</td>
</tr>
<tr>
<td>SetDItem</td>
<td>A98E</td>
</tr>
<tr>
<td>SetEmptyRgn</td>
<td>A8DD</td>
</tr>
<tr>
<td>SetEOF</td>
<td>A012</td>
</tr>
<tr>
<td>SetFileInfo</td>
<td>A00D</td>
</tr>
<tr>
<td>PBSetEOF</td>
<td>A041</td>
</tr>
<tr>
<td>PBSetFileInfo</td>
<td>A043</td>
</tr>
<tr>
<td>PBSetFilLock</td>
<td>A903</td>
</tr>
<tr>
<td>PBSetFPos</td>
<td>A044</td>
</tr>
<tr>
<td>PBSetFPos</td>
<td>A04B</td>
</tr>
<tr>
<td>PBSetHandleSize</td>
<td>A024</td>
</tr>
<tr>
<td>PBSetItemMark</td>
<td>A947</td>
</tr>
<tr>
<td>PBSetItemMark</td>
<td>A944</td>
</tr>
<tr>
<td>PBSetItemStyle</td>
<td>A942</td>
</tr>
<tr>
<td>PBSetItemStyle</td>
<td>A945</td>
</tr>
<tr>
<td>PBSetMenuBar</td>
<td>A93C</td>
</tr>
<tr>
<td>PBSetMenuFlash</td>
<td>A94A</td>
</tr>
<tr>
<td>PBSetMinCtl</td>
<td>A964</td>
</tr>
<tr>
<td>PBSetPortBits</td>
<td>A878</td>
</tr>
<tr>
<td>PBSetPortBits</td>
<td>A875</td>
</tr>
<tr>
<td>PBSetPenState</td>
<td>A899</td>
</tr>
<tr>
<td>Name</td>
<td>Trap word</td>
</tr>
<tr>
<td>--------------------</td>
<td>-----------</td>
</tr>
<tr>
<td>SetPort</td>
<td>A873</td>
</tr>
<tr>
<td>SetPt</td>
<td>A880</td>
</tr>
<tr>
<td>SetPtrSize</td>
<td>A020</td>
</tr>
<tr>
<td>SetRecRgn</td>
<td>A8DE</td>
</tr>
<tr>
<td>SetRectRgn</td>
<td></td>
</tr>
<tr>
<td>SetRect</td>
<td>A8A7</td>
</tr>
<tr>
<td>SetResAttr</td>
<td>A9A7</td>
</tr>
<tr>
<td>SetResFileAttr</td>
<td>A9F7</td>
</tr>
<tr>
<td>SetResInfo</td>
<td>A9A9</td>
</tr>
<tr>
<td>SetResLoad</td>
<td>A99B</td>
</tr>
<tr>
<td>SetResPurge</td>
<td>A993</td>
</tr>
<tr>
<td>SetStdProcs</td>
<td>A8EA</td>
</tr>
<tr>
<td>SetString</td>
<td>A907</td>
</tr>
<tr>
<td>SetTrapAddress</td>
<td>A047</td>
</tr>
<tr>
<td>SetVol</td>
<td>A015</td>
</tr>
<tr>
<td>PBSetVol</td>
<td></td>
</tr>
<tr>
<td>SetWindowPic</td>
<td>A92E</td>
</tr>
<tr>
<td>SetWRefCon</td>
<td>A918</td>
</tr>
<tr>
<td>SetWTitle</td>
<td>A91A</td>
</tr>
<tr>
<td>SetZone</td>
<td>A01B</td>
</tr>
<tr>
<td>ShieldCursor</td>
<td>A855</td>
</tr>
<tr>
<td>ShowControl</td>
<td>A957</td>
</tr>
<tr>
<td>ShowCursor</td>
<td>A853</td>
</tr>
<tr>
<td>ShowHide</td>
<td>A908</td>
</tr>
<tr>
<td>ShowPen</td>
<td>A897</td>
</tr>
<tr>
<td>ShowWindow</td>
<td>A915</td>
</tr>
<tr>
<td>SizeControl</td>
<td>A95C</td>
</tr>
<tr>
<td>SizeRsrc</td>
<td>A9A5</td>
</tr>
<tr>
<td>SizeResource</td>
<td></td>
</tr>
<tr>
<td>SizeWindow</td>
<td>A91D</td>
</tr>
<tr>
<td>SlopeFromAngle</td>
<td>A8BC</td>
</tr>
<tr>
<td>SpaceExtra</td>
<td>A88E</td>
</tr>
<tr>
<td>Status</td>
<td>A005</td>
</tr>
<tr>
<td>PBStatus</td>
<td></td>
</tr>
<tr>
<td>StdArc</td>
<td>A8BD</td>
</tr>
<tr>
<td>StdBits</td>
<td>A8EB</td>
</tr>
<tr>
<td>StdComment</td>
<td>A8F1</td>
</tr>
<tr>
<td>StdGetPic</td>
<td>A8E8</td>
</tr>
<tr>
<td>StdLine</td>
<td>A890</td>
</tr>
<tr>
<td>StdOval</td>
<td>A8B6</td>
</tr>
<tr>
<td>StdPoly</td>
<td>A8C5</td>
</tr>
<tr>
<td>StdPutPic</td>
<td>A8F0</td>
</tr>
<tr>
<td>StdRect</td>
<td>A8A0</td>
</tr>
<tr>
<td>StdRgn</td>
<td>A8D1</td>
</tr>
<tr>
<td>StdRRect</td>
<td>A8AF</td>
</tr>
<tr>
<td>StdText</td>
<td>A882</td>
</tr>
<tr>
<td>StdTxMeas</td>
<td>A8ED</td>
</tr>
<tr>
<td>StillDown</td>
<td>A973</td>
</tr>
<tr>
<td>StopAlert</td>
<td>A986</td>
</tr>
<tr>
<td>StringWidth</td>
<td>A88C</td>
</tr>
<tr>
<td>StuffHex</td>
<td>A866</td>
</tr>
</tbody>
</table>

**III-220 System Traps**
<table>
<thead>
<tr>
<th>Name</th>
<th>Trap word</th>
<th>Name</th>
<th>Trap word</th>
</tr>
</thead>
<tbody>
<tr>
<td>ValidRect</td>
<td>A92A</td>
<td>Write</td>
<td>A003</td>
</tr>
<tr>
<td>ValidRgn</td>
<td>A929</td>
<td>WriteParam</td>
<td>A038</td>
</tr>
<tr>
<td>VInstall</td>
<td>A033</td>
<td>WriteResource</td>
<td>A9B0</td>
</tr>
<tr>
<td>VRemove</td>
<td>A034</td>
<td>XorRgn</td>
<td>A8E7</td>
</tr>
<tr>
<td>WaitMouseUp</td>
<td>A977</td>
<td>ZeroScrap</td>
<td>A9FC</td>
</tr>
<tr>
<td>A000</td>
<td>Open</td>
<td>GetVol</td>
<td>A003</td>
</tr>
<tr>
<td>A001</td>
<td>Close</td>
<td>PBGetVol</td>
<td>A003</td>
</tr>
<tr>
<td>A002</td>
<td>Read</td>
<td>PBClose</td>
<td>A003</td>
</tr>
<tr>
<td>A003</td>
<td>Write</td>
<td>PBWrite</td>
<td>A003</td>
</tr>
<tr>
<td>A004</td>
<td>Control</td>
<td>PBControl</td>
<td>A003</td>
</tr>
<tr>
<td>A005</td>
<td>Status</td>
<td>PBStatus</td>
<td>A003</td>
</tr>
<tr>
<td>A006</td>
<td>KillIO</td>
<td>PBKillIO</td>
<td>A003</td>
</tr>
<tr>
<td>A007</td>
<td>GetVolInfo</td>
<td>PBGetVInfo</td>
<td>A003</td>
</tr>
<tr>
<td>A008</td>
<td>Create</td>
<td>PBCreate</td>
<td>A003</td>
</tr>
<tr>
<td>A009</td>
<td>Delete</td>
<td>PBDelete</td>
<td>A003</td>
</tr>
<tr>
<td>A00A</td>
<td>OpenRF</td>
<td>PBOpenRF</td>
<td>A003</td>
</tr>
<tr>
<td>A00B</td>
<td>Rename</td>
<td>PBRename</td>
<td>A003</td>
</tr>
<tr>
<td>A00C</td>
<td>GetFileInfo</td>
<td>PBGetInfo</td>
<td>A003</td>
</tr>
<tr>
<td>A00D</td>
<td>SetFileInfo</td>
<td>PBSetFileInfo</td>
<td>A003</td>
</tr>
<tr>
<td>A00E</td>
<td>UnmountVol</td>
<td>PBUnmountVol</td>
<td>A003</td>
</tr>
<tr>
<td>A00F</td>
<td>MountVol</td>
<td>PBMountVol</td>
<td>A003</td>
</tr>
<tr>
<td>A010</td>
<td>Allocate</td>
<td>PBAllocate</td>
<td>A003</td>
</tr>
<tr>
<td>A011</td>
<td>GetEOF</td>
<td>PBGetEOF</td>
<td>A003</td>
</tr>
<tr>
<td>A012</td>
<td>SetEOF</td>
<td>PBSetEOF</td>
<td>A003</td>
</tr>
<tr>
<td>A013</td>
<td>FlushVol</td>
<td>PBFlushVol</td>
<td>A003</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

System Traps I-221
### Inside Macintosh

<table>
<thead>
<tr>
<th>Trap word</th>
<th>Name</th>
<th>Trap word</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>A038</td>
<td>WriteParam</td>
<td>A861</td>
<td>Random</td>
</tr>
<tr>
<td>A039</td>
<td>ReadDateTime</td>
<td>A862</td>
<td>ForeColor</td>
</tr>
<tr>
<td>A03A</td>
<td>SetDateTime</td>
<td>A863</td>
<td>BackColor</td>
</tr>
<tr>
<td>A03B</td>
<td>Delay</td>
<td>A864</td>
<td>ColorBit</td>
</tr>
<tr>
<td>A03C</td>
<td>CmpString</td>
<td>A865</td>
<td>GetPixel</td>
</tr>
<tr>
<td></td>
<td>EqualString</td>
<td>A866</td>
<td>StuffHex</td>
</tr>
<tr>
<td>A03D</td>
<td>DrvrInstall</td>
<td>A867</td>
<td>LongMul</td>
</tr>
<tr>
<td></td>
<td>(internal use only)</td>
<td>A868</td>
<td>FixMul</td>
</tr>
<tr>
<td>A03E</td>
<td>DrvrRemove</td>
<td>A869</td>
<td>FixRatio</td>
</tr>
<tr>
<td></td>
<td>(internal use only)</td>
<td>A86A</td>
<td>HiWord</td>
</tr>
<tr>
<td>A03F</td>
<td>InitUtil</td>
<td>A86B</td>
<td>LoWord</td>
</tr>
<tr>
<td>A040</td>
<td>ResrvMem</td>
<td>A86C</td>
<td>FixRound</td>
</tr>
<tr>
<td>A041</td>
<td>SetFilLock</td>
<td>A86D</td>
<td>InitPort</td>
</tr>
<tr>
<td></td>
<td>PBSetFLock</td>
<td>A86E</td>
<td>InitGraf</td>
</tr>
<tr>
<td>A042</td>
<td>RstFilLock</td>
<td>A86F</td>
<td>OpenPort</td>
</tr>
<tr>
<td></td>
<td>PBRestFLock</td>
<td>A870</td>
<td>LocalToGlobal</td>
</tr>
<tr>
<td>A043</td>
<td>SetFiType</td>
<td>A871</td>
<td>GlobalToLocal</td>
</tr>
<tr>
<td></td>
<td>PBSetFVers</td>
<td>A872</td>
<td>GrafDevice</td>
</tr>
<tr>
<td>A044</td>
<td>SetFPos</td>
<td>A873</td>
<td>SetPort</td>
</tr>
<tr>
<td></td>
<td>PBSetFPos</td>
<td>A874</td>
<td>GetPort</td>
</tr>
<tr>
<td>A045</td>
<td>FlushFile</td>
<td>A875</td>
<td>SetPBBits</td>
</tr>
<tr>
<td></td>
<td>PBFlushFile</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A146</td>
<td>GetTrapAddress</td>
<td>A876</td>
<td>SetPortBits</td>
</tr>
<tr>
<td>A047</td>
<td>SetTrapAddress</td>
<td>A877</td>
<td>PortSize</td>
</tr>
<tr>
<td>A148</td>
<td>PrtZone</td>
<td>A878</td>
<td>MovePortTo</td>
</tr>
<tr>
<td>A049</td>
<td>HPurge</td>
<td>A879</td>
<td>SetOrigin</td>
</tr>
<tr>
<td>A04A</td>
<td>HNoPurge</td>
<td>A87A</td>
<td>SetClip</td>
</tr>
<tr>
<td>A04B</td>
<td>SetGrowZone</td>
<td>A87B</td>
<td>ClipRect</td>
</tr>
<tr>
<td>A04C</td>
<td>CompactMem</td>
<td>A87C</td>
<td>BackPat</td>
</tr>
<tr>
<td>A04D</td>
<td>PurgeMem</td>
<td>A87D</td>
<td>ClosePort</td>
</tr>
<tr>
<td>A04E</td>
<td>AddDrive</td>
<td>A87E</td>
<td>AddPt</td>
</tr>
<tr>
<td></td>
<td>(internal use only)</td>
<td>A87F</td>
<td>SubPt</td>
</tr>
<tr>
<td>A04F</td>
<td>RDrvrlnstall</td>
<td>A880</td>
<td>SetPt</td>
</tr>
<tr>
<td></td>
<td>(internal use only)</td>
<td>A881</td>
<td>EqualPt</td>
</tr>
<tr>
<td>A850</td>
<td>InitCursor</td>
<td>A882</td>
<td>StdText</td>
</tr>
<tr>
<td>A851</td>
<td>SetCursor</td>
<td>A883</td>
<td>DrawChar</td>
</tr>
<tr>
<td>A852</td>
<td>HideCursor</td>
<td>A884</td>
<td>DrawString</td>
</tr>
<tr>
<td>A853</td>
<td>ShowCursor</td>
<td>A885</td>
<td>DrawText</td>
</tr>
<tr>
<td>A054</td>
<td>UprString</td>
<td>A886</td>
<td>TextWidth</td>
</tr>
<tr>
<td>A855</td>
<td>ShieldCursor</td>
<td>A887</td>
<td>TextFont</td>
</tr>
<tr>
<td>A856</td>
<td>ObscureCursor</td>
<td>A888</td>
<td>TextFace</td>
</tr>
<tr>
<td>A057</td>
<td>SetAppBase</td>
<td>A889</td>
<td>TextMode</td>
</tr>
<tr>
<td></td>
<td>SetApplBase</td>
<td>A88A</td>
<td>TextSize</td>
</tr>
<tr>
<td>A858</td>
<td>BitAnd</td>
<td>A88B</td>
<td>GetFontInfo</td>
</tr>
<tr>
<td>A859</td>
<td>BitXor</td>
<td>A88C</td>
<td>StringWidth</td>
</tr>
<tr>
<td>A85A</td>
<td>BitNot</td>
<td>A88D</td>
<td>CharWidth</td>
</tr>
<tr>
<td>A85B</td>
<td>BitOr</td>
<td>A88E</td>
<td>SpaceExtra</td>
</tr>
<tr>
<td>A85C</td>
<td>BitShift</td>
<td>A890</td>
<td>StdLine</td>
</tr>
<tr>
<td>A85D</td>
<td>BitTst</td>
<td>A891</td>
<td>LineTo</td>
</tr>
<tr>
<td>A85E</td>
<td>BitSet</td>
<td>A892</td>
<td>Line</td>
</tr>
<tr>
<td>A85F</td>
<td>BitClr</td>
<td>A893</td>
<td>MoveTo</td>
</tr>
</tbody>
</table>

**III-222 System Traps**
<table>
<thead>
<tr>
<th>Trap word</th>
<th>Name</th>
<th>Trap word</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>A894</td>
<td>Move</td>
<td>A8C8</td>
<td>ErasePoly</td>
</tr>
<tr>
<td>A896</td>
<td>HidePen</td>
<td>A8C9</td>
<td>InventPoly</td>
</tr>
<tr>
<td>A897</td>
<td>ShowPen</td>
<td>A8CA</td>
<td>FillPoly</td>
</tr>
<tr>
<td>A898</td>
<td>GetPenState</td>
<td>A8CB</td>
<td>OpenPoly</td>
</tr>
<tr>
<td>A899</td>
<td>SetPenState</td>
<td>A8CC</td>
<td>ClosePgon</td>
</tr>
<tr>
<td>A89A</td>
<td>GetPen</td>
<td>A8CD</td>
<td>ClosePoly</td>
</tr>
<tr>
<td>A89B</td>
<td>PenSize</td>
<td>A8CE</td>
<td>KillPoly</td>
</tr>
<tr>
<td>A89C</td>
<td>PenMode</td>
<td>A8CF</td>
<td>OffsetPoly</td>
</tr>
<tr>
<td>A89D</td>
<td>PenPat</td>
<td>A8D0</td>
<td>PackBits</td>
</tr>
<tr>
<td>A89E</td>
<td>PenNormal</td>
<td>A8D1</td>
<td>UnPackBits</td>
</tr>
<tr>
<td>A8A0</td>
<td>StdRect</td>
<td>A8D2</td>
<td>StdRgn</td>
</tr>
<tr>
<td>A8A1</td>
<td>FrameRect</td>
<td>A8D3</td>
<td>FrameRgn</td>
</tr>
<tr>
<td>A8A2</td>
<td>PaintRect</td>
<td>A8D4</td>
<td>PaintRgn</td>
</tr>
<tr>
<td>A8A3</td>
<td>EraseRect</td>
<td>A8D5</td>
<td>EraseRgn</td>
</tr>
<tr>
<td>A8A4</td>
<td>InvertRect</td>
<td>A8D6</td>
<td>InvertRgn</td>
</tr>
<tr>
<td></td>
<td></td>
<td>A8D7</td>
<td></td>
</tr>
<tr>
<td>A8A5</td>
<td>FillRect</td>
<td>A8D8</td>
<td>NewRgn</td>
</tr>
<tr>
<td>A8A6</td>
<td>EqualRect</td>
<td>A8D9</td>
<td>DisposeRgn</td>
</tr>
<tr>
<td>A8A7</td>
<td>SetRect</td>
<td>A8DA</td>
<td>OpenRgn</td>
</tr>
<tr>
<td>A8A8</td>
<td>OffsetRect</td>
<td>A8DB</td>
<td>CloseRgn</td>
</tr>
<tr>
<td>A8A9</td>
<td>InsetRect</td>
<td>A8DC</td>
<td>CopyRgn</td>
</tr>
<tr>
<td>A8AA</td>
<td>SectRect</td>
<td>A8DD</td>
<td>SetEmptyRgn</td>
</tr>
<tr>
<td>A8AB</td>
<td>UnionRect</td>
<td>A8DE</td>
<td>SetRectRgn</td>
</tr>
<tr>
<td>A8AC</td>
<td>Pt2Rect</td>
<td>A8DF</td>
<td>RectRgn</td>
</tr>
<tr>
<td>A8AD</td>
<td>PtInRect</td>
<td>A8E0</td>
<td>OffsetRgn</td>
</tr>
<tr>
<td>A8AE</td>
<td>EmptyRect</td>
<td>A8E1</td>
<td>InsetRgn</td>
</tr>
<tr>
<td>A8AF</td>
<td>StdRRect</td>
<td>A8E2</td>
<td>EmptyRgn</td>
</tr>
<tr>
<td>A8B0</td>
<td>FrameRoundRect</td>
<td>A8E3</td>
<td>EqualRgn</td>
</tr>
<tr>
<td>A8B1</td>
<td>PaintRoundRect</td>
<td>A8E4</td>
<td>SectRgn</td>
</tr>
<tr>
<td>A8B2</td>
<td>EraseRoundRect</td>
<td>A8E5</td>
<td>UnionRgn</td>
</tr>
<tr>
<td>A8B3</td>
<td>InvertRoundRect</td>
<td>A8E6</td>
<td>DiffRgn</td>
</tr>
<tr>
<td></td>
<td></td>
<td>A8E7</td>
<td>XorRgn</td>
</tr>
<tr>
<td>A8B4</td>
<td>FillRoundRect</td>
<td>A8E8</td>
<td>PtInRgn</td>
</tr>
<tr>
<td>A8B6</td>
<td>StdOval</td>
<td>A8E9</td>
<td>RectInRgn</td>
</tr>
<tr>
<td>A8B7</td>
<td>FrameOval</td>
<td>A8EA</td>
<td>SetStdProc</td>
</tr>
<tr>
<td>A8B8</td>
<td>PaintOval</td>
<td>A8EB</td>
<td>StdBits</td>
</tr>
<tr>
<td>A8B9</td>
<td>EraseOval</td>
<td>A8EC</td>
<td>CopyBits</td>
</tr>
<tr>
<td>A8BA</td>
<td>InvertOval</td>
<td>A8ED</td>
<td>StdTxMeas</td>
</tr>
<tr>
<td>A8BB</td>
<td>FillOval</td>
<td>A8EE</td>
<td>StdGetPic</td>
</tr>
<tr>
<td>A8BC</td>
<td>SlopeFromAngle</td>
<td>A8EF</td>
<td>ScrollRect</td>
</tr>
<tr>
<td>A8BD</td>
<td>StdArc</td>
<td>A8F0</td>
<td>StdPutPic</td>
</tr>
<tr>
<td>A8BE</td>
<td>FrameArc</td>
<td>A8F1</td>
<td>StdComment</td>
</tr>
<tr>
<td>A8BF</td>
<td>PaintArc</td>
<td>A8F2</td>
<td>PicComment</td>
</tr>
<tr>
<td>A8C0</td>
<td>EraseArc</td>
<td>A8F3</td>
<td>OpenPicture</td>
</tr>
<tr>
<td>A8C1</td>
<td>InvertArc</td>
<td>A8F4</td>
<td>ClosePicture</td>
</tr>
<tr>
<td>A8C2</td>
<td>FillArc</td>
<td>A8F5</td>
<td>KillPicture</td>
</tr>
<tr>
<td>A8C3</td>
<td>PtToAngle</td>
<td>A8F6</td>
<td>DrawPicture</td>
</tr>
<tr>
<td>A8C4</td>
<td>AngleFromSlope</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A8C5</td>
<td>StdPoly</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A8C6</td>
<td>FramePoly</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A8C7</td>
<td>PaintPoly</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

System Traps III-223
Inside Macintosh

<table>
<thead>
<tr>
<th>Trap word</th>
<th>Name</th>
<th>Trap word</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>A8F8</td>
<td>ScalePt</td>
<td>A929</td>
<td>ValidRgn</td>
</tr>
<tr>
<td>A8F9</td>
<td>MapPt</td>
<td>A92A</td>
<td>ValidRect</td>
</tr>
<tr>
<td>A8FA</td>
<td>MapRect</td>
<td>A92B</td>
<td>GrowWindow</td>
</tr>
<tr>
<td>A8FB</td>
<td>MapRgn</td>
<td>A92C</td>
<td>FindWindow</td>
</tr>
<tr>
<td>A8FC</td>
<td>MapPoly</td>
<td>A92D</td>
<td>CloseWindow</td>
</tr>
<tr>
<td>A8FE</td>
<td>InitFonts</td>
<td>A92E</td>
<td>SetWindowPic</td>
</tr>
<tr>
<td>A8FF</td>
<td>GetFName</td>
<td>A92F</td>
<td>GetWindowPic</td>
</tr>
<tr>
<td>A900</td>
<td>GetFNum</td>
<td>A930</td>
<td>InitMenus</td>
</tr>
<tr>
<td>A901</td>
<td>FSMswapFont</td>
<td>A931</td>
<td>NewMenu</td>
</tr>
<tr>
<td>A902</td>
<td>RealFont</td>
<td>A932</td>
<td>DisposeMenu</td>
</tr>
<tr>
<td>A903</td>
<td>SetFontLock</td>
<td>A933</td>
<td>DisposeMenu</td>
</tr>
<tr>
<td>A904</td>
<td>DrawGrowIcon</td>
<td>A934</td>
<td>AppendMenu</td>
</tr>
<tr>
<td>A905</td>
<td>DragGrayRgn</td>
<td>A935</td>
<td>ClearMenuBar</td>
</tr>
<tr>
<td>A906</td>
<td>NewString</td>
<td>A936</td>
<td>InsertMenu</td>
</tr>
<tr>
<td>A907</td>
<td>SetString</td>
<td>A937</td>
<td>DeleteMenu</td>
</tr>
<tr>
<td>A908</td>
<td>ShowHide</td>
<td>A938</td>
<td>DrawMenuBar</td>
</tr>
<tr>
<td>A909</td>
<td>CalcVis</td>
<td>A939</td>
<td>HiliteMenu</td>
</tr>
<tr>
<td>A90A</td>
<td>CalcVisBehind</td>
<td>A93A</td>
<td>EnableItem</td>
</tr>
<tr>
<td>A90B</td>
<td>ClipAbove</td>
<td>A93B</td>
<td>DisableItem</td>
</tr>
<tr>
<td>A90C</td>
<td>PaintOne</td>
<td>A93C</td>
<td>GetMenuBar</td>
</tr>
<tr>
<td>A90D</td>
<td>PaintBehind</td>
<td>A93D</td>
<td>SetMenuBar</td>
</tr>
<tr>
<td>A90E</td>
<td>SaveOld</td>
<td>A93E</td>
<td>MenuSelect</td>
</tr>
<tr>
<td>A90F</td>
<td>DrawNew</td>
<td>A93F</td>
<td>MenuKey</td>
</tr>
<tr>
<td>A910</td>
<td>GetWMgrPort</td>
<td>A940</td>
<td>GetltemIcon</td>
</tr>
<tr>
<td>A911</td>
<td>CheckUpdate</td>
<td>A941</td>
<td>SetItemIcon</td>
</tr>
<tr>
<td>A912</td>
<td>InitWindows</td>
<td>A942</td>
<td>GetltemStyle</td>
</tr>
<tr>
<td>A913</td>
<td>NewWindow</td>
<td>A943</td>
<td>SetItemStyle</td>
</tr>
<tr>
<td>A914</td>
<td>DisposWindow</td>
<td>A944</td>
<td>SetItemStyle</td>
</tr>
<tr>
<td>A915</td>
<td>ShowWindow</td>
<td>A945</td>
<td>GetltemMark</td>
</tr>
<tr>
<td>A916</td>
<td>HideWindow</td>
<td>A946</td>
<td>GetltemMark</td>
</tr>
<tr>
<td>A917</td>
<td>GetWRefCon</td>
<td>A947</td>
<td>GetltemMark</td>
</tr>
<tr>
<td>A918</td>
<td>SetWRefCon</td>
<td>A948</td>
<td>GetltemMark</td>
</tr>
<tr>
<td>A919</td>
<td>GetWTitle</td>
<td>A949</td>
<td>GetltemMark</td>
</tr>
<tr>
<td>A91A</td>
<td>SetWTitle</td>
<td>A94A</td>
<td>GetltemMark</td>
</tr>
<tr>
<td>A91B</td>
<td>MoveWindow</td>
<td>A94B</td>
<td>GetltemMark</td>
</tr>
<tr>
<td>A91C</td>
<td>HiliteWindow</td>
<td>A94C</td>
<td>GetltemMark</td>
</tr>
<tr>
<td>A91D</td>
<td>SizeWindow</td>
<td>A94D</td>
<td>GetltemMark</td>
</tr>
<tr>
<td>A91E</td>
<td>TrackGoAway</td>
<td>A94E</td>
<td>GetltemMark</td>
</tr>
<tr>
<td>A91F</td>
<td>SelectWindow</td>
<td>A94F</td>
<td>GetltemMark</td>
</tr>
<tr>
<td>A920</td>
<td>BringToFront</td>
<td>A950</td>
<td>GetltemMark</td>
</tr>
<tr>
<td>A921</td>
<td>SendBehind</td>
<td>A951</td>
<td>GetltemMark</td>
</tr>
<tr>
<td>A922</td>
<td>BeginUpdate</td>
<td>A952</td>
<td>GetltemMark</td>
</tr>
<tr>
<td>A923</td>
<td>EndUpdate</td>
<td>A953</td>
<td>GetltemMark</td>
</tr>
<tr>
<td>A924</td>
<td>FrontWindow</td>
<td>A954</td>
<td>GetltemMark</td>
</tr>
<tr>
<td>A925</td>
<td>DragWindow</td>
<td>A955</td>
<td>GetltemMark</td>
</tr>
<tr>
<td>A926</td>
<td>DragTheRgn</td>
<td>A956</td>
<td>GetltemMark</td>
</tr>
<tr>
<td>A927</td>
<td>InvalRgn</td>
<td>A957</td>
<td>GetltemMark</td>
</tr>
<tr>
<td>A928</td>
<td>InvalRect</td>
<td>A958</td>
<td>GetltemMark</td>
</tr>
</tbody>
</table>

III-224 System Traps
<table>
<thead>
<tr>
<th>Trap word</th>
<th>Name</th>
<th>Trap word</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>A955</td>
<td>DisposeControl</td>
<td>A986</td>
<td>StopAlert</td>
</tr>
<tr>
<td>A956</td>
<td>KillControls</td>
<td>A987</td>
<td>NoteAlert</td>
</tr>
<tr>
<td>A957</td>
<td>ShowControl</td>
<td>A988</td>
<td>CautionAlert</td>
</tr>
<tr>
<td>A958</td>
<td>HideControl</td>
<td>A989</td>
<td>CouldAlert</td>
</tr>
<tr>
<td>A959</td>
<td>MoveControl</td>
<td>A98A</td>
<td>FreeAlert</td>
</tr>
<tr>
<td>A95A</td>
<td>GetCRefCon</td>
<td>A98B</td>
<td>ParamText</td>
</tr>
<tr>
<td>A95B</td>
<td>SetCRefCon</td>
<td>A98C</td>
<td>ErrorSound</td>
</tr>
<tr>
<td>A95C</td>
<td>SizeControl</td>
<td>A98D</td>
<td>GetDlgItem</td>
</tr>
<tr>
<td>A95D</td>
<td>HiliteControl</td>
<td>A98E</td>
<td>SetDlgItem</td>
</tr>
<tr>
<td>A95E</td>
<td>GetCTitle</td>
<td>A98F</td>
<td>SetIText</td>
</tr>
<tr>
<td>A95F</td>
<td>SetCTitle</td>
<td>A98G</td>
<td>GetIText</td>
</tr>
<tr>
<td>A960</td>
<td>GetCtlValue</td>
<td>A98H</td>
<td>ModalDialog</td>
</tr>
<tr>
<td>A961</td>
<td>GetMinCtl</td>
<td>A98I</td>
<td>DetachResource</td>
</tr>
<tr>
<td>A962</td>
<td>GetMaxCtl</td>
<td>A98J</td>
<td>SetResPurge</td>
</tr>
<tr>
<td>A963</td>
<td>SetCtlValue</td>
<td>A98K</td>
<td>CurResFile</td>
</tr>
<tr>
<td>A964</td>
<td>SetMinCtl</td>
<td>A98L</td>
<td>InitResources</td>
</tr>
<tr>
<td>A965</td>
<td>SetMaxCtl</td>
<td>A98M</td>
<td>RsRefZoneInit</td>
</tr>
<tr>
<td>A966</td>
<td>TestControl</td>
<td>A98N</td>
<td>OpenResFile</td>
</tr>
<tr>
<td>A967</td>
<td>DragControl</td>
<td>A98O</td>
<td>UseResFile</td>
</tr>
<tr>
<td>A968</td>
<td>TrackControl</td>
<td>A98P</td>
<td>UpdateResFile</td>
</tr>
<tr>
<td>A969</td>
<td>DrawControls</td>
<td>A98Q</td>
<td>CloseResFile</td>
</tr>
<tr>
<td>A96A</td>
<td>GetCtlAction</td>
<td>A98R</td>
<td>SetResLoad</td>
</tr>
<tr>
<td>A96B</td>
<td>SetCtlAction</td>
<td>A98S</td>
<td>CountResources</td>
</tr>
<tr>
<td>A96C</td>
<td>FindControl</td>
<td>A98T</td>
<td>GetIndResource</td>
</tr>
<tr>
<td>A96E</td>
<td>Dequeue</td>
<td>A98U</td>
<td>CountTypes</td>
</tr>
<tr>
<td>A96F</td>
<td>Enqueue</td>
<td>A98V</td>
<td>GetIndType</td>
</tr>
<tr>
<td>A970</td>
<td>GetNextEvent</td>
<td>A98W</td>
<td>GetResource</td>
</tr>
<tr>
<td>A971</td>
<td>EventAvail</td>
<td>A98X</td>
<td>GetNameResource</td>
</tr>
<tr>
<td>A972</td>
<td>GetMouse</td>
<td>A98Y</td>
<td>LoadResource</td>
</tr>
<tr>
<td>A973</td>
<td>StillDown</td>
<td>A98Z</td>
<td>ReleaseResource</td>
</tr>
<tr>
<td>A974</td>
<td>Button</td>
<td>A990</td>
<td>HomeResFile</td>
</tr>
<tr>
<td>A975</td>
<td>TickCount</td>
<td>A991</td>
<td>SizeRsRef</td>
</tr>
<tr>
<td>A976</td>
<td>GetKeys</td>
<td>A992</td>
<td>GetResAttrs</td>
</tr>
<tr>
<td>A977</td>
<td>WaitMouseUp</td>
<td>A993</td>
<td>SetResAttr</td>
</tr>
<tr>
<td>A979</td>
<td>CouldDialog</td>
<td>A994</td>
<td>GetResInfo</td>
</tr>
<tr>
<td>A97A</td>
<td>FreeDialog</td>
<td>A995</td>
<td>SetResInfo</td>
</tr>
<tr>
<td>A97B</td>
<td>InitDialogs</td>
<td>A996</td>
<td>ChangedResource</td>
</tr>
<tr>
<td>A97C</td>
<td>GetNewDialog</td>
<td>A997</td>
<td>AddResource</td>
</tr>
<tr>
<td>A97D</td>
<td>NewDialog</td>
<td>A998</td>
<td>RmveResource</td>
</tr>
<tr>
<td>A97E</td>
<td>SetIText</td>
<td>A999</td>
<td>ResError</td>
</tr>
<tr>
<td>A97F</td>
<td>IsDialogEvent</td>
<td>A99A</td>
<td>WriteResource</td>
</tr>
<tr>
<td>A980</td>
<td>DialogSelect</td>
<td>A99B</td>
<td>CreateResFile</td>
</tr>
<tr>
<td>A981</td>
<td>DrawDialog</td>
<td>A99C</td>
<td>SystemEvent</td>
</tr>
<tr>
<td>A982</td>
<td>CloseDialog</td>
<td>A99D</td>
<td>SystemClick</td>
</tr>
<tr>
<td>A983</td>
<td>DisposDialog</td>
<td>A99E</td>
<td>SystemTask</td>
</tr>
<tr>
<td>A985</td>
<td>Alert</td>
<td>A99F</td>
<td>SystemMenu</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

System Traps III-225
<table>
<thead>
<tr>
<th>Trap word</th>
<th>Name</th>
<th>Trap word</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>A9BA</td>
<td>GetString</td>
<td>A9E9</td>
<td>Pack2</td>
</tr>
<tr>
<td>A9BB</td>
<td>GetIcon</td>
<td></td>
<td>DIBadMount (0)</td>
</tr>
<tr>
<td>A9BC</td>
<td>GetPicture</td>
<td></td>
<td>DILoad (2)</td>
</tr>
<tr>
<td>A9BD</td>
<td>GetNewWindow</td>
<td></td>
<td>DIUnload (4)</td>
</tr>
<tr>
<td>A9BE</td>
<td>GetNewControl</td>
<td></td>
<td>DIFormat (6)</td>
</tr>
<tr>
<td>A9BF</td>
<td>GetMenu</td>
<td></td>
<td>DVERIFY (8)</td>
</tr>
<tr>
<td>A9C0</td>
<td>GetNewMBar</td>
<td>A9EA</td>
<td>DIZero (10)</td>
</tr>
<tr>
<td>A9C1</td>
<td>UniqueID</td>
<td></td>
<td>Pack3</td>
</tr>
<tr>
<td>A9C2</td>
<td>SysEdit</td>
<td></td>
<td>SFPutFile (1)</td>
</tr>
<tr>
<td>A9C6</td>
<td>Secs2Date</td>
<td>A9EB</td>
<td>Pack4</td>
</tr>
<tr>
<td>A9C7</td>
<td>Date2Secs</td>
<td></td>
<td>(synonym: FP68K)</td>
</tr>
<tr>
<td>A9C8</td>
<td>SysBeep</td>
<td>A9EC</td>
<td>Pack5</td>
</tr>
<tr>
<td>A9C9</td>
<td>SysError</td>
<td></td>
<td>(synonym: Elems68K)</td>
</tr>
<tr>
<td>A9CB</td>
<td>TEGetText</td>
<td>A9ED</td>
<td>Pack6</td>
</tr>
<tr>
<td>A9CC</td>
<td>TElinit</td>
<td></td>
<td>IUDateString (0)</td>
</tr>
<tr>
<td>A9CD</td>
<td>TEDispose</td>
<td></td>
<td>IUTimeString (2)</td>
</tr>
<tr>
<td>A9CE</td>
<td>TextBox</td>
<td></td>
<td>IUMetric (4)</td>
</tr>
<tr>
<td>A9CF</td>
<td>TESerText</td>
<td></td>
<td>IUDGetIntl (6)</td>
</tr>
<tr>
<td>A9D0</td>
<td>TECalText</td>
<td></td>
<td>IUSetInt (8)</td>
</tr>
<tr>
<td>A9D1</td>
<td>TESetSelect</td>
<td></td>
<td>IUMagIntl (10)</td>
</tr>
<tr>
<td>A9D2</td>
<td>TENew</td>
<td></td>
<td>IUMagString (12)</td>
</tr>
<tr>
<td>A9D3</td>
<td>TEOUpdate</td>
<td></td>
<td>IUDatePString (14)</td>
</tr>
<tr>
<td>A9D4</td>
<td>TEClick</td>
<td></td>
<td>IUTimePSTRING (16)</td>
</tr>
<tr>
<td>A9D5</td>
<td>TECopy</td>
<td>A9EE</td>
<td>Pack7</td>
</tr>
<tr>
<td>A9D6</td>
<td>TECut</td>
<td></td>
<td>NumToString (0)</td>
</tr>
<tr>
<td>A9D7</td>
<td>TEDelete</td>
<td></td>
<td>StringToNum (1)</td>
</tr>
<tr>
<td>A9D8</td>
<td>TEActivate</td>
<td></td>
<td>PersAndHand</td>
</tr>
<tr>
<td>A9D9</td>
<td>TEDeactivate</td>
<td>A9EF</td>
<td>LoadSeg</td>
</tr>
<tr>
<td>A9DA</td>
<td>TEIdle</td>
<td>A9F0</td>
<td>UnloadSeg</td>
</tr>
<tr>
<td>A9DB</td>
<td>TEPaste</td>
<td>A9F1</td>
<td>Launch</td>
</tr>
<tr>
<td>A9DC</td>
<td>TKEY</td>
<td>A9F2</td>
<td>Chain</td>
</tr>
<tr>
<td>A9DD</td>
<td>TEScroll</td>
<td>A9F3</td>
<td>ExitToShell</td>
</tr>
<tr>
<td>A9DE</td>
<td>TELink</td>
<td>A9F4</td>
<td>GetAppParams</td>
</tr>
<tr>
<td>A9DF</td>
<td>TESetJust</td>
<td>A9F5</td>
<td>GetResFileAttrs</td>
</tr>
<tr>
<td>A9E0</td>
<td>Munger</td>
<td>A9F6</td>
<td>InfoScrap</td>
</tr>
<tr>
<td>A9E1</td>
<td>HandToHand</td>
<td>A9F7</td>
<td>UnlodeResFileAttr</td>
</tr>
<tr>
<td>A9E2</td>
<td>PTrToXHand</td>
<td>A9F9</td>
<td>LodeResFileAttr</td>
</tr>
<tr>
<td>A9E3</td>
<td>PTrToHand</td>
<td>A9FA</td>
<td>UnlodeScrap</td>
</tr>
<tr>
<td>A9E4</td>
<td>HandAndHand</td>
<td>A9FB</td>
<td>LodeScrap</td>
</tr>
<tr>
<td>A9E5</td>
<td>InitPack</td>
<td></td>
<td>LoadScrap</td>
</tr>
<tr>
<td>A9E6</td>
<td>InitAllPacks</td>
<td></td>
<td>ZeroScrap</td>
</tr>
<tr>
<td>A9E7</td>
<td>Pack0</td>
<td>A9FC</td>
<td>GetScrap</td>
</tr>
<tr>
<td>A9E8</td>
<td>Pack1</td>
<td>A9FD</td>
<td>PutScrap</td>
</tr>
</tbody>
</table>

**III-226 System Traps**
### APPENDIX D: GLOBAL VARIABLES

This appendix gives an alphabetical list of all system global variables described in *Inside Macintosh*, along with their locations in memory.

<table>
<thead>
<tr>
<th>Name</th>
<th>Location</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABusVars</td>
<td>$2D8</td>
<td>Pointer to AppleTalk variables</td>
</tr>
<tr>
<td>ACount</td>
<td>$A9A</td>
<td>Stage number (0 through 3) of last alert (word)</td>
</tr>
<tr>
<td>ANumber</td>
<td>$A98</td>
<td>Resource ID of last alert (word)</td>
</tr>
<tr>
<td>ApFontID</td>
<td>$984</td>
<td>Font number of application font (word)</td>
</tr>
<tr>
<td>ApplLimit</td>
<td>$130</td>
<td>Application heap limit</td>
</tr>
<tr>
<td>ApplScratch</td>
<td>$A78</td>
<td>12-byte scratch area reserved for use by applications</td>
</tr>
<tr>
<td>ApplZone</td>
<td>$2AA</td>
<td>Address of application heap zone</td>
</tr>
<tr>
<td>AppParmHandle</td>
<td>$AEC</td>
<td>Handle to Finder information</td>
</tr>
<tr>
<td>BufPtr</td>
<td>$10C</td>
<td>Address of end of jump table</td>
</tr>
<tr>
<td>BufTgDate</td>
<td>$304</td>
<td>File tags buffer: date and time of last modification (long)</td>
</tr>
<tr>
<td>BufTgFBkNum</td>
<td>$302</td>
<td>File tags buffer: logical block number (word)</td>
</tr>
<tr>
<td>BufTgFFlg</td>
<td>$300</td>
<td>File tags buffer: flags (word: bit 1=1 if resource fork)</td>
</tr>
<tr>
<td>BufTgFNum</td>
<td>$2FC</td>
<td>File tags buffer: file number (long)</td>
</tr>
<tr>
<td>CaretTime</td>
<td>$2F4</td>
<td>Caret-blink interval in ticks (long)</td>
</tr>
<tr>
<td>CrsrThresh</td>
<td>$8EC</td>
<td>Mouse-scaling threshold (word)</td>
</tr>
<tr>
<td>CurActivate</td>
<td>$A64</td>
<td>Pointer to window to receive activate event</td>
</tr>
<tr>
<td>CurApName</td>
<td>$910</td>
<td>Name of current application (length byte followed by up to 31 characters)</td>
</tr>
<tr>
<td>CurApRefNum</td>
<td>$900</td>
<td>Reference number of current application's resource file (word)</td>
</tr>
<tr>
<td>CurDeactive</td>
<td>$A68</td>
<td>Pointer to window to receive deactivate event</td>
</tr>
<tr>
<td>CurJTOffset</td>
<td>$934</td>
<td>Offset to jump table from location pointed to by A5 (word)</td>
</tr>
<tr>
<td>CurMap</td>
<td>$A5A</td>
<td>Reference number of current resource file (word)</td>
</tr>
<tr>
<td>CurPageOption</td>
<td>$936</td>
<td>Sound/screen buffer configuration passed to Chain or Launch (word)</td>
</tr>
<tr>
<td>CurPitch</td>
<td>$280</td>
<td>Value of count in square-wave synthesizer buffer (word)</td>
</tr>
<tr>
<td>CurrentA5</td>
<td>$904</td>
<td>Address of boundary between application globals and application parameters</td>
</tr>
<tr>
<td>CurStackBase</td>
<td>$908</td>
<td>Address of base of stack; start of application globals</td>
</tr>
<tr>
<td>DABeeper</td>
<td>$A9C</td>
<td>Address of current sound procedure</td>
</tr>
<tr>
<td>DAStrings</td>
<td>$AA0</td>
<td>Handles to ParamText strings (16 bytes)</td>
</tr>
<tr>
<td>Name</td>
<td>Location</td>
<td>Contents</td>
</tr>
<tr>
<td>-----------------------</td>
<td>----------</td>
<td>--------------------------------------------------------------------------</td>
</tr>
<tr>
<td>DefltStack</td>
<td>$322</td>
<td>Default space allotment for stack (long)</td>
</tr>
<tr>
<td>DefvCBPtr</td>
<td>$352</td>
<td>Pointer to default volume control block</td>
</tr>
<tr>
<td>DeskHook</td>
<td>$A6C</td>
<td>Address of procedure for painting desktop or responding to clicks on desktop</td>
</tr>
<tr>
<td>DeskPattern</td>
<td>$A3C</td>
<td>Pattern with which desktop is painted (8 bytes)</td>
</tr>
<tr>
<td>DlgFont</td>
<td>$AFA</td>
<td>Font number for dialogs and alerts (word)</td>
</tr>
<tr>
<td>DoubleTime</td>
<td>$2F0</td>
<td>Double-click interval in ticks (long)</td>
</tr>
<tr>
<td>DragHook</td>
<td>$9F6</td>
<td>Address of procedure to execute during TrackGoAway, DragWindow, GrowWindow, DragGrayRgn, TrackControl, and DragControl</td>
</tr>
<tr>
<td>DragPattern</td>
<td>$A34</td>
<td>Pattern of dragged region's outline (8 bytes)</td>
</tr>
<tr>
<td>DrvQHdr</td>
<td>$308</td>
<td>Drive queue header (10 bytes)</td>
</tr>
<tr>
<td>DSAAlertRect</td>
<td>$3F8</td>
<td>Rectangle enclosing system error alert (8 bytes)</td>
</tr>
<tr>
<td>DSAAlertTab</td>
<td>$2BA</td>
<td>Pointer to system error alert table in use</td>
</tr>
<tr>
<td>DSErrCode</td>
<td>$AF0</td>
<td>Current system error ID (word)</td>
</tr>
<tr>
<td>EventQueue</td>
<td>$14A</td>
<td>Event queue header (10 bytes)</td>
</tr>
<tr>
<td>ExtStsDT</td>
<td>$2BE</td>
<td>External/status interrupt vector table (16 bytes)</td>
</tr>
<tr>
<td>FCBSPtr</td>
<td>$34E</td>
<td>Pointer to file-control-block buffer</td>
</tr>
<tr>
<td>FinderName</td>
<td>$2E0</td>
<td>Name of the Finder (length byte followed by up to 15 characters)</td>
</tr>
<tr>
<td>FScaleDisable</td>
<td>$A63</td>
<td>Nonzero to disable font scaling (byte)</td>
</tr>
<tr>
<td>FSQHdr</td>
<td>$360</td>
<td>File I/O queue header (10 bytes)</td>
</tr>
<tr>
<td>GhostWindow</td>
<td>$A84</td>
<td>Pointer to window never to be considered frontmost</td>
</tr>
<tr>
<td>GrayRgn</td>
<td>$9EE</td>
<td>Handle to region drawn as desktop</td>
</tr>
<tr>
<td>GZRootHnd</td>
<td>$328</td>
<td>Handle to relocatable block not to be moved by grow zone function</td>
</tr>
<tr>
<td>HeapEnd</td>
<td>$114</td>
<td>Address of end of application heap zone</td>
</tr>
<tr>
<td>JFetch</td>
<td>$8F4</td>
<td>Jump vector for Fetch function</td>
</tr>
<tr>
<td>JIODone</td>
<td>$8FC</td>
<td>Jump vector for IODone function</td>
</tr>
<tr>
<td>JournalFlag</td>
<td>$8DE</td>
<td>Journaling mode (word)</td>
</tr>
<tr>
<td>JournalRef</td>
<td>$8E8</td>
<td>Reference number of journaling device driver (word)</td>
</tr>
<tr>
<td>JStash</td>
<td>$8F8</td>
<td>Jump vector for Stash function</td>
</tr>
<tr>
<td>KeyRepThresh</td>
<td>$190</td>
<td>Auto-key rate (word)</td>
</tr>
<tr>
<td>KeyThresh</td>
<td>$18E</td>
<td>Auto-key threshold (word)</td>
</tr>
<tr>
<td>Lo3Bytes</td>
<td>$31A</td>
<td>$00FFFFFF</td>
</tr>
<tr>
<td>Lvl1DT</td>
<td>$192</td>
<td>Level-1 secondary interrupt vector table (32 bytes)</td>
</tr>
<tr>
<td>Name</td>
<td>Location</td>
<td>Contents</td>
</tr>
<tr>
<td>------------------</td>
<td>----------</td>
<td>--------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Lvl2DT</td>
<td>$1B2</td>
<td>Level-2 secondary interrupt vector table (32 bytes)</td>
</tr>
<tr>
<td>MBarEnable</td>
<td>$A20</td>
<td>Unique menu ID for active desk accessory, when menu bar belongs to the accessory (word)</td>
</tr>
<tr>
<td>MBarHook</td>
<td>$A2C</td>
<td>Address of routine called by MenuSelect before menu is drawn</td>
</tr>
<tr>
<td>MemTop</td>
<td>$108</td>
<td>Address of end of RAM (on Macintosh XL, end of RAM available to applications)</td>
</tr>
<tr>
<td>MenuFlash</td>
<td>$A24</td>
<td>Count for duration of menu item blinking (word)</td>
</tr>
<tr>
<td>MenuHook</td>
<td>$A30</td>
<td>Address of routine called during MenuSelect</td>
</tr>
<tr>
<td>MenuList</td>
<td>$A1C</td>
<td>Handle to current menu list</td>
</tr>
<tr>
<td>MinStack</td>
<td>$31E</td>
<td>Minimum space allotment for stack (long)</td>
</tr>
<tr>
<td>MinusOne</td>
<td>$A06</td>
<td>$FFFFFFFF</td>
</tr>
<tr>
<td>OldContent</td>
<td>$9EA</td>
<td>Handle to saved content region</td>
</tr>
<tr>
<td>OldStructure</td>
<td>$9E6</td>
<td>Handle to saved structure region</td>
</tr>
<tr>
<td>OneOne</td>
<td>$A02</td>
<td>$00010001</td>
</tr>
<tr>
<td>PaintWhite</td>
<td>$9DC</td>
<td>Flag for whether to paint window white before update event (word)</td>
</tr>
<tr>
<td>PortBUse</td>
<td>$291</td>
<td>Current availability of serial port B (byte)</td>
</tr>
<tr>
<td>PrintErr</td>
<td>$944</td>
<td>Result code from last Printing Manager routine (word)</td>
</tr>
<tr>
<td>RAMBase</td>
<td>$2B2</td>
<td>Trap dispatch table's base address for routines in RAM</td>
</tr>
<tr>
<td>ResErr</td>
<td>$A60</td>
<td>Current value of ResError (word)</td>
</tr>
<tr>
<td>ResErrProc</td>
<td>$AF2</td>
<td>Address of resource error procedure</td>
</tr>
<tr>
<td>ResLoad</td>
<td>$A5E</td>
<td>Current SetResLoad state (word)</td>
</tr>
<tr>
<td>ResumeProc</td>
<td>$A8C</td>
<td>Address of resume procedure</td>
</tr>
<tr>
<td>RndSeed</td>
<td>$156</td>
<td>Random number seed (long)</td>
</tr>
<tr>
<td>ROMBase</td>
<td>$2AE</td>
<td>Base address of ROM</td>
</tr>
<tr>
<td>ROMFont0</td>
<td>$980</td>
<td>Handle to font record for system font</td>
</tr>
<tr>
<td>SaveUpdate</td>
<td>$9DA</td>
<td>Flag for whether to generate update events (word)</td>
</tr>
<tr>
<td>SaveVisRgn</td>
<td>$9F2</td>
<td>Handle to saved visRgn</td>
</tr>
<tr>
<td>SCCRd</td>
<td>$1D8</td>
<td>SCC read base address</td>
</tr>
<tr>
<td>SCCWr</td>
<td>$1DC</td>
<td>SCC write base address</td>
</tr>
<tr>
<td>ScrapCount</td>
<td>$968</td>
<td>Count changed by ZeroScrap (word)</td>
</tr>
<tr>
<td>ScrapHandle</td>
<td>$964</td>
<td>Handle to desk scrap in memory</td>
</tr>
<tr>
<td>ScrapName</td>
<td>$96C</td>
<td>Pointer to scrap file name (preceded by length byte)</td>
</tr>
<tr>
<td>ScrapSize</td>
<td>$960</td>
<td>Size in bytes of desk scrap (long)</td>
</tr>
<tr>
<td>ScrapState</td>
<td>$96A</td>
<td>Tells where desk scrap is (word)</td>
</tr>
<tr>
<td>Name</td>
<td>Location</td>
<td>Contents</td>
</tr>
<tr>
<td>-----------------</td>
<td>----------</td>
<td>--------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Scratch8</td>
<td>$9FA</td>
<td>8-byte scratch area</td>
</tr>
<tr>
<td>Scratch20</td>
<td>$1E4</td>
<td>20-byte scratch area</td>
</tr>
<tr>
<td>ScrDmpEnb</td>
<td>$2F8</td>
<td>0 if GetNextEvent shouldn't process Command-Shift-number combinations (byte)</td>
</tr>
<tr>
<td>ScrHRes</td>
<td>$104</td>
<td>Pixels per inch horizontally (word)</td>
</tr>
<tr>
<td>ScrnBase</td>
<td>$824</td>
<td>Address of main screen buffer</td>
</tr>
<tr>
<td>ScrVRes</td>
<td>$102</td>
<td>Pixels per inch vertically (word)</td>
</tr>
<tr>
<td>SdVolume</td>
<td>$260</td>
<td>Current speaker volume (byte: low-order three bits only)</td>
</tr>
<tr>
<td>SEvtEnb</td>
<td>$15C</td>
<td>0 if SystemEvent should return FALSE (byte)</td>
</tr>
<tr>
<td>SFSaveDisk</td>
<td>$214</td>
<td>Negative of volume reference number used by Standard File Package (word)</td>
</tr>
<tr>
<td>SoundBase</td>
<td>$266</td>
<td>Pointer to free-form synthesizer buffer</td>
</tr>
<tr>
<td>SoundLevel</td>
<td>$27F</td>
<td>Amplitude in 740-byte buffer (byte)</td>
</tr>
<tr>
<td>SoundPtr</td>
<td>$262</td>
<td>Pointer to four-tone record</td>
</tr>
<tr>
<td>SPAAlarm</td>
<td>$200</td>
<td>Alarm setting (long)</td>
</tr>
<tr>
<td>SPATalkA</td>
<td>$1F9</td>
<td>AppleTalk node ID hint for modem port (byte)</td>
</tr>
<tr>
<td>SPATalkB</td>
<td>$1FA</td>
<td>AppleTalk node ID hint for printer port (byte)</td>
</tr>
<tr>
<td>SPClikCaret</td>
<td>$209</td>
<td>Double-click and caret-blink times (byte)</td>
</tr>
<tr>
<td>SPConfig</td>
<td>$1FB</td>
<td>Use types for serial ports (byte)</td>
</tr>
<tr>
<td>SPFont</td>
<td>$204</td>
<td>Application font number minus 1 (word)</td>
</tr>
<tr>
<td>SPKbd</td>
<td>$206</td>
<td>Auto-key threshold and rate (byte)</td>
</tr>
<tr>
<td>SPMisc2</td>
<td>$20B</td>
<td>Mouse scaling, system startup disk, menu blink (byte)</td>
</tr>
<tr>
<td>SPPortA</td>
<td>$1FC</td>
<td>Modem port configuration (word)</td>
</tr>
<tr>
<td>SPPortB</td>
<td>$1FE</td>
<td>Printer port configuration (word)</td>
</tr>
<tr>
<td>SPPrint</td>
<td>$207</td>
<td>Printer connection (byte)</td>
</tr>
<tr>
<td>SPValid</td>
<td>$1F8</td>
<td>Validity status (byte)</td>
</tr>
<tr>
<td>SPVolCtl</td>
<td>$208</td>
<td>Speaker volume setting in parameter RAM (byte)</td>
</tr>
<tr>
<td>SysEvtMask</td>
<td>$144</td>
<td>System event mask (word)</td>
</tr>
<tr>
<td>SysMap</td>
<td>$A58</td>
<td>Reference number of system resource file (word)</td>
</tr>
<tr>
<td>SysMapHndl</td>
<td>$A54</td>
<td>Handle to map of system resource file</td>
</tr>
<tr>
<td>SysParam</td>
<td>$1F8</td>
<td>Low-memory copy of parameter RAM (20 bytes)</td>
</tr>
<tr>
<td>SysResName</td>
<td>$AD8</td>
<td>Name of system resource file (length byte followed by up to 19 characters)</td>
</tr>
<tr>
<td>SysZone</td>
<td>$2A6</td>
<td>Address of system heap zone</td>
</tr>
<tr>
<td>TEDoText</td>
<td>$A70</td>
<td>Address of TextEdit multi-purpose routine</td>
</tr>
</tbody>
</table>

*III-230 Global Variables*
<table>
<thead>
<tr>
<th>Name</th>
<th>Location</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>TERecal</td>
<td>$A74</td>
<td>Address of routine to recalculate line starts for TextEdit</td>
</tr>
<tr>
<td>TEScrpHandle</td>
<td>$AB4</td>
<td>Handle to TextEdit scrap</td>
</tr>
<tr>
<td>TEScrpLength</td>
<td>$AB0</td>
<td>Size in bytes of TextEdit scrap (long)</td>
</tr>
<tr>
<td>TheMenu</td>
<td>$A26</td>
<td>Menu ID of currently highlighted menu (word)</td>
</tr>
<tr>
<td>TheZone</td>
<td>$118</td>
<td>Address of current heap zone</td>
</tr>
<tr>
<td>Ticks</td>
<td>$16A</td>
<td>Current number of ticks since system startup (long)</td>
</tr>
<tr>
<td>Time</td>
<td>$20C</td>
<td>Seconds since midnight, January 1, 1904 (long)</td>
</tr>
<tr>
<td>ToExtFS</td>
<td>$3F2</td>
<td>Pointer to external file system</td>
</tr>
<tr>
<td>ToolScratch</td>
<td>$9CE</td>
<td>8-byte scratch area</td>
</tr>
<tr>
<td>TopMapHndl</td>
<td>$A50</td>
<td>Handle to resource map of most recently opened resource file</td>
</tr>
<tr>
<td>UTableBase</td>
<td>$11C</td>
<td>Base address of unit table</td>
</tr>
<tr>
<td>VBLQueue</td>
<td>$160</td>
<td>Vertical retrace queue header (10 bytes)</td>
</tr>
<tr>
<td>VCBQHdr</td>
<td>$356</td>
<td>Volume-control-block queue header (10 bytes)</td>
</tr>
<tr>
<td>VIA</td>
<td>$1DA</td>
<td>VIA base address</td>
</tr>
<tr>
<td>WindowList</td>
<td>$9D6</td>
<td>Pointer to first window in window list; 0 if using events but not windows</td>
</tr>
<tr>
<td>WMgrPort</td>
<td>$9DE</td>
<td>Pointer to Window Manager port</td>
</tr>
</tbody>
</table>

*Global Variables*
GLOSSARY

access path: A description of the route that the File Manager follows to access a file; created when a file is opened.

access path buffer: Memory used by the File Manager to transfer data between an application and a file.

action procedure: A procedure, used by the Control Manager function TrackControl, that defines an action to be performed repeatedly for as long as the mouse button is held down.

activate event: An event generated by the Window Manager when a window changes from active to inactive or vice versa.

active control: A control that will respond to the user's actions with the mouse.

active window: The frontmost window on the desktop.

address mark: In a sector, information that's used internally by the Disk Driver, including information it uses to determine the position of the sector on the disk.


ALAP frame: A packet of data transmitted and received by ALAP.

ALAP protocol type: An identifier used to match particular kinds of packets with a particular protocol handler.

alert: A warning or report of an error, in the form of an alert box, sound from the Macintosh's speaker, or both.

alert box: A box that appears on the screen to give a warning or report an error during a Macintosh application.

alert template: A resource that contains information from which the Dialog Manager can create an alert.

alert window: The window in which an alert box is displayed.

alias: A different name for the same entity.

allocate: To reserve an area of memory for use.

allocation block: Volume space composed of an integral number of logical blocks.

amplitude: The maximum vertical distance of a periodic wave from the horizontal line about which the wave oscillates.

AppleTalk address: A socket's number and its node ID number.

AppleTalk Manager: An interface to a pair of RAM device drivers that enable programs to send and receive information via an AppleTalk network.

AppleTalk Transaction Protocol (ATP): An AppleTalk protocol that's a DDP client. It allows one ATP client to request another ATP client to perform some activity and report the activity's result as a response to the requesting socket with guaranteed delivery.

application font: The font your application will use unless you specify otherwise—Geneva, by default.

application heap: The portion of the heap available to the running application program and the Toolbox.

application heap limit: The boundary between the space available for the application heap and the space available for the stack.

application heap zone: The heap zone initially provided by the Memory Manager for use by the application program and the Toolbox; initially equivalent to the application heap, but may be subdivided into two or more independent heap zones.

application parameters: Thirty-two bytes of memory, located above the application globals, reserved for system use. The first application parameter is the address of the first QuickDraw global variable.

application space: Memory that's available for dynamic allocation by applications.

application window: A window created as the result of something done by the application, either directly or indirectly (as through the Dialog Manager).

ascent: The vertical distance from a font's base line to its ascent line.

ascent line: A horizontal line that coincides with the tops of the tallest characters in a font.

asynchronous communication: A method of data transmission where the receiving and sending devices don't share a common timer, and no timing data is transmitted.

asynchronous execution: After calling a routine asynchronously, an application is free to perform other tasks until the routine is completed.

at-least-once transaction: An ATP transaction in which the requested operation is performed at least once, and possibly several times.


auto-key event: An event generated repeatedly when the user presses and holds down a character key on the keyboard or keypad.

auto-key rate: The rate at which a character key repeats after it's begun to do so.
auto-key threshold: The length of time a character key must be held down before it begins to repeat.

background procedure: A procedure passed to the Printing Manager to be run during idle times in the printing process.

base line: A horizontal line that coincides with the bottom of each character in a font, excluding descenders (such as the tail of a "p").

baud rate: The measure of the total number of bits sent over a transmission line per second.

Binary-Decimal Conversion Package: A Macintosh package for converting integers to decimal strings and vice versa.

bit image: A collection of bits in memory that have a rectilinear representation. The screen is a visible bit image.

bit map: A set of bits that represent the position and state of a corresponding set of items; in QuickDraw, a pointer to a bit image, the row width of that image, and its boundary rectangle.

block: A group regarded as a unit; usually refers to data or memory in which data is stored. See allocation block and memory block.

block contents: The area that's available for use in a memory block.

block device: A device that reads and writes blocks of bytes at a time. It can read or write any accessible block on demand.

block header: The internal "housekeeping" information maintained by the Memory Manager at the beginning of each block in a heap zone.

block map: Same as volume allocation block map.

boundary rectangle: A rectangle, defined as part of a QuickDraw bit map, that encloses the active area of the bit image and imposes a coordinate system on it. Its top left corner is always aligned around the first bit in the bit image.

break: The condition resulting when a device maintains its transmission line in the space state for at least one frame.

bridge: An intelligent link between two or more AppleTalk networks.

broadcast service: An ALAP service in which a frame is sent to all nodes on an AppleTalk network.

bundle: A resource that maps local IDs of resources to their actual resource IDs; used to provide mappings for file references and icon lists needed by the Finder.

button: A standard Macintosh control that causes some immediate or continuous action when clicked or pressed with the mouse. See also radio button.
caret: A generic term meaning a symbol that indicates where something should be inserted in text. The specific symbol used is a vertical bar (|).

caret-blink time: The interval between blinks of the caret that marks an insertion point.

character code: An integer representing the character that a key or combination of keys on the keyboard or keypad stands for.

character device: A device that reads or writes a stream of characters, one at a time. It can neither skip characters nor go back to a previous character.

character image: An arrangement of bits that defines a character in a font.

character key: A key that generates a keyboard event when pressed; any key except Shift, Caps Lock, Command, or Option.

character offset: The horizontal separation between a character rectangle and a font rectangle.

character origin: The point on a base line used as a reference location for drawing a character.

character position: An index into an array containing text, starting at 0 for the first character.

character rectangle: A rectangle enclosing an entire character image. Its sides are defined by the image width and the font height.

character style: A set of stylistic variations, such as bold, italic, and underline. The empty set indicates plain text (no stylistic variations).

character width: The distance to move the pen from one character's origin to the next character's origin.

check box: A standard Macintosh control that displays a setting, either checked (on) or unchecked (off). Clicking inside a check box reverses its setting.

clipping: Limiting drawing to within the bounds of a particular area.

clipping region: Same as clipRgn.

clipRgn: The region to which an application limits drawing in a grafPort.

clock chip: A special chip in which are stored parameter RAM and the current setting for the date and time. This chip is powered by a battery when the system is off, thus preserving the information.

close routine: The part of a device driver's code that implements Device Manager Close calls.

closed driver: A device driver that cannot be read from or written to.

closed file: A file without an access path. Closed files cannot be read from or written to.

compaction: The process of moving allocated blocks within a heap zone in order to collect the free space into a single block.
completion routine: Any application-defined code to be executed when an asynchronous call to a routine is completed.

content region: The area of a window that the application draws in.

control: An object in a window on the Macintosh screen with which the user, using the mouse, can cause instant action with visible results or change settings to modify a future action.

control definition function: A function called by the Control Manager when it needs to perform type-dependent operations on a particular type of control, such as drawing the control.

control definition ID: A number passed to control-creation routines to indicate the type of control. It consists of the control definition function's resource ID and a variation code.

control information: Information transmitted by an application to a device driver. It may select modes of operation, start or stop processes, enable buffers, choose protocols, and so on.

control list: A list of all the controls associated with a given window.

Control Manager: The part of the Toolbox that provides routines for creating and manipulating controls (such as buttons, check boxes, and scroll bars).

control record: The internal representation of a control, where the Control Manager stores all the information it needs for its operations on that control.

control routine: The part of a device driver's code that implements Device Manager Control and KillIO calls.

control template: A resource that contains information from which the Control Manager can create a control.

coordinate plane: A two-dimensional grid. In QuickDraw, the grid coordinates are integers ranging from -32767 to 32767, and all grid lines are infinitely thin.

current heap zone: The heap zone currently under attention, to which most Memory Manager operations implicitly apply.

current resource file: The last resource file opened, unless you specify otherwise with a Resource Manager routine.

cursor: A 16-by-16 bit image that appears on the screen and is controlled by the mouse; called the "pointer" in Macintosh user manuals.

cursor level: A value, initialized by InitCursor, that keeps track of the number of times the cursor has been hidden.

data bits: Data communications bits that encode transmitted characters.

data buffer: Heap space containing information to be written to a file or device driver from an application, or read from a file or device driver to an application.

data fork: The part of a file that contains data accessed via the File Manager.
data mark: In a sector, information that primarily contains data from an application.

Datagram: A packet of data transmitted by DDP.

Datagram Delivery Protocol (DDP): An AppleTalk protocol that's an ALAP client, managing socket-to-socket delivery of datagrams over AppleTalk internets.

date/time record: An alternate representation of the date and time (which is stored on the clock chip in seconds since midnight, January 1, 1904).

DDP: See Datagram Delivery Protocol.

default button: In an alert box or modal dialog, the button whose effect will occur if the user presses Return or Enter. In an alert box, it's boldly outlined; in a modal dialog, it's boldly outlined or the OK button.

default volume: A volume that will receive I/O during a File Manager routine call, whenever no other volume is specified.

dereference: To refer to a block by its master pointer instead of its handle.

descent: The vertical distance from a font's base line to its descent line.

descent line: A horizontal line that coincides with the bottoms of the characters in a font that extend furthest below the base line.

desk accessory: A "mini-application", implemented as a device driver, that can be run at the same time as a Macintosh application.

Desk Manager: The part of the Toolbox that supports the use of desk accessories from an application.

desk scrap: The place where data is stored when it's cut (or copied) and pasted among applications and desk accessories.

desktop: The screen as a surface for doing work on the Macintosh.

Desktop file: A resource file in which the Finder stores the version data, bundle, icons, and file references for each application on the volume.

destination rectangle: In TextEdit, the rectangle in which the text is drawn.

device: A part of the Macintosh, or a piece of external equipment, that can transfer information into or out of the Macintosh.

device control entry: A 40-byte relocatable block of heap space that tells the Device Manager the location of a driver's routines, the location of a driver's I/O queue, and other information.

device driver: A program that controls the exchange of information between an application and a device.

device driver event: An event generated by one of the Macintosh's device drivers.
Device Manager: The part of the Operating System that supports device I/O.

dial: A control with a moving indicator that displays a quantitative setting or value. Depending on the type of dial, the user may be able to change the setting by dragging the indicator with the mouse.

dialog: Same as dialog box.

dialog box: A box that a Macintosh application displays to request information it needs to complete a command, or to report that it's waiting for a process to complete.

Dialog Manager: The part of the Toolbox that provides routines for implementing dialogs and alerts.

dialog record: The internal representation of a dialog, where the Dialog Manager stores all the information it needs for its operations on that dialog.

dialog template: A resource that contains information from which the Dialog Manager can create a dialog.

dialog window: The window in which a dialog box is displayed.

dimmed: Drawn in gray rather than black

disabled: A disabled menu item or menu is one that cannot be chosen; the menu item or menu title appears dimmed. A disabled item in a dialog or alert box has no effect when clicked.

Disk Driver: The device driver that controls data storage and retrieval on 3 1/2-inch disks.

Disk Initialization Package: A Macintosh package for initializing and naming new disks; called by the Standard File Package.

disk-inserted event: An event generated when the user inserts a disk in a disk drive or takes any other action that requires a volume to be mounted.

display rectangle: A rectangle that determines where an item is displayed within a dialog or alert box.

document window: The standard Macintosh window for presenting a document.

double-click time: The greatest interval between a mouse-up and mouse-down event that would qualify two mouse clicks as a double-click.

draft printing: Printing a document immediately as it's drawn in the printing grafPort.

drag region: A region in a window frame. Dragging inside this region moves the window to a new location and makes it the active window unless the Command key was down.

drive number: A number used to identify a disk drive. The internal drive is number 1, the external drive is number 2, and any additional drives will have larger numbers.

drive queue: A list of disk drives connected to the Macintosh.
driver name: A sequence of up to 255 printing characters used to refer to an open device driver. Driver names always begin with a period (.).

driver I/O queue: A queue containing the parameter blocks of all I/O requests for one device driver.

driver reference number: A number from -1 to -32 that uniquely identifies an individual device driver.
edit record: A complete editing environment in TextEdit, which includes the text to be edited, the grafPort and rectangle in which to display the text, the arrangement of the text within the rectangle, and other editing and display information.

empty handle: A handle that points to a NIL master pointer, signifying that the underlying relocatable block has been purged.

empty shape: A shape that contains no bits, such as one defined by only a single point.
end-of-file: See logical end-of-file or physical end-of-file.

entity name: An identifier for an entity, of the form object:type@zone.

event: A notification to an application of some occurrence that the application may want to respond to.

event code: An integer representing a particular type of event.

Event Manager: See Toolbox Event Manager or Operating System Event Manager.

event mask: A parameter passed to an Event Manager routine to specify which types of events the routine should apply to.

event message: A field of an event record containing information specific to the particular type of event.

event queue: The Operating System Event Manager's list of pending events.

event record: The internal representation of an event, through which your program learns all pertinent information about that event.

exactly-once transaction: An ATP transaction in which the requested operation is performed only once.

exception: An error or abnormal condition detected by the processor in the course of program execution; includes interrupts and traps.

exception vector: One of 64 vectors in low memory that point to the routines that are to get control in the event of an exception.

external reference: A reference to a routine or variable defined in a separate compilation or assembly.
file: A named, ordered sequence of bytes; a principal means by which data is stored and transmitted on the Macintosh.

file control block: A fixed-length data structure, contained in the file-control-block buffer, where information about an access path is stored.

file-control-block buffer: A nonrelocatable block in the system heap that contains one file control block for each access path.

file directory: The part of a volume that contains descriptions and locations of all the files on the volume.

file I/O queue: A queue containing parameter blocks for all I/O requests to the File Manager.

File Manager: The part of the Operating System that supports file I/O.

file name: A sequence of up to 255 printing characters, excluding colons (:), that identifies a file.

file number: A unique number assigned to a file, which the File Manager uses to distinguish it from other files on the volume. A file number specifies the file's entry in a file directory.

file reference: A resource that provides the Finder with file and icon information about an application.

file tags: Information associated with each logical block, designed to allow reconstruction of files on a volume whose directory or other file-access information has been destroyed.

file tags buffer: A location in memory where file tags are read from and written to.

file type: A four-character sequence, specified when a file is created, that identifies the type of file.

Finder information: Information that the Finder provides to an application upon starting it up, telling it which documents to open or print.

fixed-point number: A signed 32-bit quantity containing an integer part in the high-order word and a fractional part in the low-order word.

fixed-width font: A font whose characters all have the same width.

Floating-Point Arithmetic Package: A Macintosh package that supports extended-precision arithmetic according to IEEE Standard 754.

font: The complete set of characters of one typeface.

font characterization table: A table of parameters in a device driver that specifies how best to adapt fonts to that device.

font height: The vertical distance from a font's ascent line to its descent line.
Font Manager: The part of the Toolbox that supports the use of various character fonts for QuickDraw when it draws text.

font number: The number by which you identify a font to QuickDraw or the Font Manager.

font record: A data structure that contains all the information describing a font.

font rectangle: The smallest rectangle enclosing all the character images in a font, if the images were all superimposed over the same character origin.

font size: The size of a font in points; equivalent to the distance between the ascent line of one line of text and the ascent line of the next line of single-spaced text.

fork: One of the two parts of a file; see data fork and resource fork.

four-tone record: A data structure describing the tones produced by a four-tone synthesizer.

four-tone synthesizer: The part of the Sound Driver used to make simple harmonic tones, with up to four "voices" producing sound simultaneously.

frame: The time elapsed from the start bit to the last stop bit during serial communication.

frame check sequence: A 16-bit value generated by the AppleTalk hardware, used by the receiving node to detect transmission errors.

frame header: Information at the beginning of a packet.

frame pointer: A pointer to the end of the local variables within a routine's stack frame, held in an address register and manipulated with the LINK and UNLK instructions.

frame trailer: Information at the end of an ALAP frame.

framed shape: A shape that's drawn outlined and hollow.

framing error: The condition resulting when a device doesn't receive a stop bit when expected.

free block: A memory block containing space available for allocation.

free-form synthesizer: The part of the Sound Driver used to make complex music and speech.

frequency: The number of cycles per second (also called hertz) at which a wave oscillates.

full-duplex communication: A method of data transmission where two devices transmit data simultaneously.

global coordinate system: The coordinate system based on the top left corner of the bit image being at (0,0).

go-away region: A region in a window frame. Clicking inside this region of the active window makes the window close or disappear.
grafPort: A complete drawing environment, including such elements as a bit map, a subset of it in which to draw, a character font, patterns for drawing and erasing, and other pen characteristics.

grow image: The image pulled around when the user drags inside the grow region; whatever is appropriate to show that the window's size will change.

grow region: A window region, usually within the content region, where dragging changes the size of an active window.

grow zone function: A function supplied by the application program to help the Memory Manager create free space within a heap zone.

handle: A pointer to a master pointer, which designates a relocatable block in the heap by double indirection.

hardware overrun error: The condition that occurs when the SCC's buffer becomes full.

heap: The area of memory in which space is dynamically allocated and released on demand, using the Memory Manager.

heap zone: An area of memory initialized by the Memory Manager for heap allocation.

highlight: To display an object on the screen in a distinctive visual way, such as inverting it.

horizontal blanking interval: The time between the display of the rightmost pixel on one line and the leftmost pixel on the next line.

hotSpot: The point in a cursor that's aligned with the mouse location.

icon: A 32-by-32 bit image that graphically represents an object, concept, or message.

icon list: A resource consisting of a list of icons.

icon number: A digit from 1 to 255 to which the Menu Manager adds 256 to get the resource ID of an icon associated with a menu item.

image width: The width of a character image.

inactive control: A control that won't respond to the user's actions with the mouse. An inactive control is highlighted in some special way, such as dimmed.

inactive window: Any window that isn't the frontmost window on the desktop.

indicator: The moving part of a dial that displays its current setting.

input driver: A device driver that receives serial data via a serial port and transfers it to an application.

insertion point: An empty selection range; the character position where text will be inserted (usually marked with a blinking caret).
interface routine: A routine called from Pascal whose purpose is to trap to a certain Toolbox or Operating System routine.

International Utilities Package: A Macintosh package that gives you access to country-dependent information such as the formats for numbers, currency, dates, and times.


internet address: The AppleTalk address and network number of a socket.

interrupt: An exception that's signaled to the processor by a device, to notify the processor of a change in condition of the device, such as the completion of an I/O request.

interrupt handler: A routine that services interrupts.

interrupt priority level: A number identifying the importance of the interrupt. It indicates which device is interrupting, and which interrupt handler should be executed.

interrupt vector: A pointer to an interrupt handler.

invert: To highlight by changing white pixels to black and vice versa.

invisible control: A control that's not drawn in its window.

invisible window: A window that's not drawn in its plane on the desktop.

I/O queue: See driver I/O queue or file I/O queue.

I/O request: A request for input from or output to a file or device driver; caused by calling a File Manager or Device Manager routine asynchronously.

item: In dialog and alert boxes, a control, icon, picture, or piece of text, each displayed inside its own display rectangle. See also menu item.

item list: A list of information about all the items in a dialog or alert box.

item number: The index, starting from 1, of an item in an item list.

IWM: "Integrated Woz Machine"; the custom chip that controls the 3 1/2-inch disk drives.

job dialog: A dialog that sets information about one printing job; associated with the Print command.

journal code: A code passed by a Toolbox Event Manager routine in its Control call to the journaling device driver, to designate which routine is making the Control call.

journaling mechanism: A mechanism that allows you to feed the Toolbox Event Manager events from some source other than the user.

jump table: A table that contains one entry for every routine in an application and is the means by which the loading and unloading of segments is implemented.
justification: The horizontal placement of lines of text relative to the edges of the rectangle in which the text is drawn.

kern: To draw part of a character so that it overlaps an adjacent character.

key code: An integer representing a key on the keyboard or keypad, without reference to the character that the key stands for.

key-down event: An event generated when the user presses a character key on the keyboard or keypad.

key-up event: An event generated when the user releases a character key on the keyboard or keypad.

keyboard configuration: A resource that defines a particular keyboard layout by associating a character code with each key or combination of keys on the keyboard or keypad.

keyboard equivalent: The combination of the Command key and another key, used to invoke a menu item from the keyboard.

keyboard event: An event generated when the user presses, releases, or holds down a character key on the keyboard or keypad; any key-down, key-up, or auto-key event.

leading: The amount of blank vertical space between the descent line of one line of text and the ascent line of the next line of single-spaced text.

ligature: A character that combines two letters.

list separator: The character that separates numbers, as when a list of numbers is entered by the user.

local coordinate system: The coordinate system local to a grafPort, imposed by the boundary rectangle defined in its bit map.

local ID: A number that refers to an icon list or file reference in an application's resource file and is mapped to an actual resource ID by a bundle.

location table: An array of words (one for each character in a font) that specifies the location of each character's image in the font's bit image.

lock: To temporarily prevent a relocatable block from being moved during heap compaction.

lock bit: A bit in the master pointer to a relocatable block that indicates whether the block is currently locked.

locked file: A file whose data cannot be changed.

locked volume: A volume whose data cannot be changed. Volumes can be locked by either a software flag or a hardware setting.

logical block: Volume space composed of 512 consecutive bytes of standard information and an additional number of bytes of information specific to the Disk Driver.
logical end-of-file: The position of one byte past the last byte in a file; equal to the actual number of bytes in the file.

logical size: The number of bytes in a memory block's contents.

magnitude: The vertical distance between any given point on a wave and the horizontal line about which the wave oscillates.

main event loop: In a standard Macintosh application program, a loop that repeatedly calls the Toolbox Event Manager to get events and then responds to them as appropriate.

main segment: The segment containing the main program.

mark: The position of the next byte in a file that will be read or written.

mark state: The state of a transmission line indicating a binary 1.

master directory block: Part of the data structure of a volume; contains the volume information and the volume allocation block map.

master pointer: A single pointer to a relocatable block, maintained by the Memory Manager and updated whenever the block is moved, purged, or reallocated. All handles to a relocatable block refer to it by double indirection through the master pointer.

memory block: An area of contiguous memory within a heap zone.

Memory Manager: The part of the Operating System that dynamically allocates and releases memory space in the heap.

menu: A list of menu items that appears when the user points to a menu title in the menu bar and presses the mouse button. Dragging through the menu and releasing over an enabled menu item chooses that item.

menu bar: The horizontal strip at the top of the Macintosh screen that contains the menu titles of all menus in the menu list.

menu definition procedure: A procedure called by the Menu Manager when it needs to perform type-dependent operations on a particular type of menu, such as drawing the menu.

menu ID: A number in the menu record that identifies the menu.

menu item: A choice in a menu, usually a command to the current application.

menu item number: The index, starting from 1, of a menu item in a menu.

menu list: A list containing menu handles for all menus in the menu bar, along with information on the position of each menu.

Menu Manager: The part of the Toolbox that deals with setting up menus and letting the user choose from them.
menu record: The internal representation of a menu, where the Menu Manager stores all the information it needs for its operations on that menu.

menu title: A word or phrase in the menu bar that designates one menu.

missing symbol: A character to be drawn in case of a request to draw a character that’s missing from a particular font.

modal dialog: A dialog that requires the user to respond before doing any other work on the desktop.

modeless dialog: A dialog that allows the user to work elsewhere on the desktop before responding.

modifier key: A key (Shift, Caps Lock, Option, or Command) that generates no keyboard events of its own, but changes the meaning of other keys or mouse actions.

mounted volume: A volume that previously was inserted into a disk drive and had descriptive information read from it by the File Manager.

mouse-down event: An event generated when the user presses the mouse button.

mouse scaling: A feature that causes the cursor to move twice as far during a mouse stroke than it would have otherwise, provided the change in the cursor’s position exceeds the mouse-scaling threshold within one tick after the mouse is moved.

mouse-scaling threshold: A number of pixels which, if exceeded by the sum of the horizontal and vertical changes in the cursor position during one tick of mouse movement, causes mouse scaling to occur (if that feature is turned on); normally six pixels.

mouse-up event: An event generated when the user releases the mouse button.

Name-Binding Protocol (NBP): An AppleTalk protocol that’s a DDP client, used to convert entity names to their internet socket addresses.

name lookup: An NBP operation that allows clients to obtain the internet addresses of entities from their names.

names directory: The union of all name tables in an internet.

names information socket: The socket in a node used to implement NBP (always socket number 2).

names table: A list of each entity’s name and internet address in a node.

NBP: See Name-Binding Protocol.

NBP tuple: An entity name and an internet address.

network event: An event generated by the AppleTalk Manager.

network number: An identifier for an AppleTalk network.
network-visible entity: A named socket client on an internet.

newline character: Any character, but usually Return (ASCII code $0D), that indicates the end of a sequence of bytes.

newline mode: A mode of reading data where the end of the data is indicated by a newline character (and not by a specific byte count).

node: A device that's attached to and communicates via an AppleTalk network.

node ID: A number, dynamically assigned, that identifies a node.

nonbreaking space: The character with ASCII code $CA; drawn as a space the same width as a digit, but interpreted as a nonblank character for the purposes of word wraparound and selection.

nonrelocatable block: A block whose location in the heap is fixed and can't be moved during heap compaction.

null event: An event reported when there are no other events to report.

off-line volume: A mounted volume with all but 94 bytes of its descriptive information released.

offset/width table: An array of words that specifies the character offsets and character widths of all characters in a font.

on-line volume: A mounted volume with its volume buffer and descriptive information contained in memory.

open driver: A driver that can be read from and written to.

open file: A file with an access path. Open files can be read from and written to.

open permission: Information about a file that indicates whether the file can be read from, written to, or both.

open routine: The part of a device driver's code that implements Device Manager Open calls.

Operating System: The lowest-level software in the Macintosh. It does basic tasks such as I/O, memory management, and interrupt handling.

Operating System Event Manager: The part of the Operating System that reports hardware-related events such as mouse-button presses and keystrokes.

Operating System Utilities: Operating System routines that perform miscellaneous tasks such as getting the date and time, finding out the user's preferred speaker volume and other preferences, and doing simple string comparison.

output driver: A device driver that receives data via a serial port and transfers it to an application.
overrun error: See hardware overrun error and software overrun error.

package: A set of routines and data types that's stored as a resource and brought into memory only when needed.

Package Manager: The part of the Toolbox that lets you access Macintosh RAM-based packages.

page rectangle: The rectangle marking the boundaries of a printed page image. The boundary rectangle, portRect, and clipRgn of the printing grafPort are set to this rectangle.

palette: A collection of small symbols, usually enclosed in rectangles, that represent operations and can be selected by the user.

pane: An independently scrollable area of a window, for showing a different part of the same document.

panel: An area of a window that shows a different interpretation of the same part of a document.

paper rectangle: The rectangle marking the boundaries of the physical sheet of paper on which a page is printed.

parameter block: A data structure used to transfer information between applications and certain Operating System routines.

parameter RAM: In the clock chip, 20 bytes where settings such as those made with the Control Panel desk accessory are preserved.

parity bit: A data communications bit used to verify that data bits received by a device match the data bits transmitted by another device.

parity error: The condition resulting when the parity bit received by a device isn't what was expected.

part code: An integer between 1 and 253 that stands for a particular part of a control (possibly the entire control).

path reference number: A number that uniquely identifies an individual access path; assigned when the access path is created.

pattern: An 8-by-8 bit image, used to define a repeating design (such as stripes) or tone (such as gray).

pattern transfer mode: One of eight transfer modes for drawing lines or shapes with a pattern.

period: The time elapsed during one complete cycle of a wave.

phase: Some fraction of a wave cycle (measured from a fixed point on the wave).

physical end-of-file: The position of one byte past the last allocation block of a file; equal to 1 more than the maximum number of bytes the file can contain.
physical size: The actual number of bytes a memory block occupies within its heap zone.

picture: A saved sequence of QuickDraw drawing commands (and, optionally, picture comments) that you can play back later with a single procedure call; also, the image resulting from these commands.

picture comments: Data stored in the definition of a picture that doesn't affect the picture's appearance but may be used to provide additional information about the picture when it's played back.

picture frame: A rectangle, defined as part of a picture, that surrounds the picture and gives a frame of reference for scaling when the picture is played back.

pixel: The visual representation of a bit on the screen (white if the bit is 0, black if it's 1).

plane: The front-to-back position of a window on the desktop.

point: The intersection of a horizontal grid line and a vertical grid line on the coordinate plane, defined by a horizontal and a vertical coordinate; also, a typographical term meaning approximately 1/72 inch.

polygon: A sequence of connected lines, defined by QuickDraw line-drawing commands.

port: See grafPort.

portBits: The bit map of a grafPort.

portRect: A rectangle, defined as part of a grafPort, that encloses a subset of the bit map for use by the grafPort.

post: To place an event in the event queue for later processing.

prime routine: The part of a device driver's code that implements Device Manager Read and Write calls.

print record: A record containing all the information needed by the Printing Manager to perform a particular printing job.

Printer Driver: The device driver for the currently installed printer.

printer resource file: A file containing all the resources needed to run the Printing Manager with a particular printer.

printing grafPort: A special grafPort customized for printing instead of drawing on the screen.

Printing Manager: The routines and data types that enable applications to communicate with the Printer Driver to print on any variety of printer via the same interface.

processor priority: Bits 8-10 of the MC68000's status register, indicating which interrupts will be processed and which will be ignored.
proportional font: A font whose characters all have character widths that are proportional to their image width.

protocol: A well-defined set of communications rules.

protocol handler: A software process in a node that recognizes different kinds of frames by their ALAP type and services them.

protocol handler table: A list of the protocol handlers for a node.

purge: To remove a relocatable block from the heap, leaving its master pointer allocated but set to NIL.

purge bit: A bit in the master pointer to a relocatable block that indicates whether the block is currently purgeable.

purge warning procedure: A procedure associated with a particular heap zone that's called whenever a block is purged from that zone.

purgeable block: A relocatable block that can be purged from the heap.

queue: A list of identically structured entries linked together by pointers.

QuickDraw: The part of the Toolbox that performs all graphic operations on the Macintosh screen.

radio button: A standard Macintosh control that displays a setting, either on or off, and is part of a group in which only one button can be on at a time.

RAM: The Macintosh's random access memory, which contains exception vectors, buffers used by hardware devices, the system and application heaps, the stack, and other information used by applications.

read/write permission: Information associated with an access path that indicates whether the file can be read from, written to, both read from and written to, or whatever the file's open permission allows.

reallocate: To allocate new space in the heap for a purged block, updating its master pointer to point to its new location.

reference number: A number greater than 0, returned by the Resource Manager when a resource file is opened, by which you can refer to that file. In Resource Manager routines that expect a reference number, 0 represents the system resource file.

reference value: In a window record or control record, a 32-bit field that an application program may store into and access for any purpose.

region: An arbitrary area or set of areas on the QuickDraw coordinate plane. The outline of a region should be one or more closed loops.

register-based routine: A Toolbox or Operating System routine that receives its parameters and returns its results, if any, in registers.
relative handle: A handle to a relocatable block expressed as the offset of its master pointer within the heap zone, rather than as the absolute memory address of the master pointer.

release: To free an allocated area of memory, making it available for reuse.

release timer: A timer for determining when an exactly-once response buffer can be released.

relocatable block: A block that can be moved within the heap during compaction.

resource: Data or code stored in a resource file and managed by the Resource Manager.

resource attribute: One of several characteristics, specified by bits in a resource reference, that determine how the resource should be dealt with.

resource data: In a resource file, the data that comprises a resource.

resource file: The resource fork of a file.

resource fork: The part of a file that contains data used by an application (such as menus, fonts, and icons). The resource fork of an application file also contains the application code itself.

resource header: At the beginning of a resource file, data that gives the offsets to and lengths of the resource data and resource map.

resource ID: A number that, together with the resource type, identifies a resource in a resource file. Every resource has an ID number.

Resource Manager: The part of the Toolbox that reads and writes resources.

resource map: In a resource file, data that is read into memory when the file is opened and that, given a resource specification, leads to the corresponding resource data.

resource name: A string that, together with the resource type, identifies a resource in a resource file. A resource may or may not have a name.

resource reference: In a resource map, an entry that identifies a resource and contains either an offset to its resource data in the resource file or a handle to the data if it's already been read into memory.

resource specification: A resource type and either a resource ID or a resource name.

resource type: The type of a resource in a resource file, designated by a sequence of four characters (such as 'MENU' for a menu).

response BDS: A data structure used to pass response information to the ATP module.

result code: An integer indicating whether a routine completed its task successfully or was prevented by some error condition (or other special condition, such as reaching the end of a file).

resume procedure: A procedure within an application that allows the application to recover from system errors.
retry count: The maximum number of retransmissions for an NBP or ATP packet.

retry interval: The time between retransmissions of a packet by NBP or ATP.

ROM: The Macintosh's permanent read-only memory, which contains the routines for the Toolbox and Operating System, and the various system traps.

routine selector: An integer that's pushed onto the stack before the _PackN macro is invoked, to identify which routine to execute. (N is the resource ID of a package; all macros for calling routines in the package expand to invoke _PackN.)

routing table: A table in a bridge that contains routing information.

Routing Table Maintenance Protocol (RTMP): An AppleTalk protocol that's used internally by AppleTalk to maintain tables for routing datagrams through an internet.

row width: The number of bytes in each row of a bit image.

RTMP: See Routing Table Maintenance Protocol.

RTMP socket: The socket in a node used to implement RTMP.

RTMP stub: The RTMP code in a nonbridge node.

scaling factor: A value, given as a fraction, that specifies the amount a character should be stretched or shrunk before it's drawn.

SCC: See Serial Communications Controller.

scrap: A place where cut or copied data is stored.

scrap file: The file containing the desk scrap (usually named "Clipboard File").

Scrap Manager: The part of the Toolbox that enables cutting and pasting between applications, desk accessories, or an application and a desk accessory.

screen buffer: A block of memory from which the video display reads the information to be displayed.

sector: Disk space composed of 512 consecutive bytes of standard information and 12 bytes of file tags.

segment: One of several parts into which the code of an application may be divided. Not all segments need to be in memory at the same time.

Segment Loader: The part of the Operating System that loads the code of an application into memory, either as a single unit or divided into dynamically loaded segments.

selection range: The series of characters (inversely highlighted), or the character position (marked with a blinking caret), at which the next editing operation will occur.
sequence number: A number from 0 to 7, assigned to an ATP response datagram to indicate its ordering within the response.

Serial Communications Controller (SCC): The chip that handles serial I/O through the modem and printer ports.

serial data: Data communicated over a single-path communication line, one bit at a time.

Serial Driver: A device driver that controls communication, via serial ports, between applications and serial peripheral devices.

signature: A four-character sequence that uniquely identifies an application to the Finder.

socket: A logical entity within the node of a network.

socket client: A software process in a node that owns a socket.

socket listener: The portion of a socket client that receives and services datagrams addressed to that socket.

socket number: An identifier for a socket.

socket table: A listing of all the socket listeners for each active socket in a node.

software overrun error: The condition that occurs when an input driver's buffer becomes full.

solid shape: A shape that's filled in with any pattern.

sound buffer: A block of memory from which the sound generator reads the information to create an audio waveform.

Sound Driver: The device driver that controls sound generation in an application.

sound procedure: A procedure associated with an alert that will emit one of up to four sounds from the Macintosh's speaker. Its integer parameter ranges from 0 to 3 and specifies which sound.

source transfer mode: One of eight transfer modes for drawing text or transferring any bit image between two bit maps.

space state: The state of a transmission line indicating a binary 0.

spool printing: Writing a representation of a document's printed image to disk or to memory, and then printing it (as opposed to immediate draft printing).

square-wave synthesizer: The part of the Sound Driver used to produce less harmonic sounds than the four-tone synthesizer, such as beeps.

stack: The area of memory in which space is allocated and released in LIFO (last-in-first-out) order.
stack-based routine: A Toolbox or Operating System routine that receives its parameters and returns its results, if any, on the stack.

stack frame: The area of the stack used by a routine for its parameters, return address, local variables, and temporary storage.

stage: Every alert has four stages, corresponding to consecutive occurrences of the alert, and a different response may be specified for each stage.

Standard File Package: A Macintosh package for presenting the standard user interface when a file is to be saved or opened.

start bit: A serial data communications bit that signals that the next bits transmitted are data bits.

status information: Information transmitted to an application by a device driver. It may indicate the current mode of operation, the readiness of the device, the occurrence of errors, and so on.

status routine: The part of a device driver's code that implements Device Manager Status calls.

stop bit: A serial data communications bit that signals the end of data bits.

structure region: An entire window; its complete "structure".

style: See character style.

style dialog: A dialog that sets options affecting the page dimensions; associated with the Page Setup command.

synchronous execution: After calling a routine synchronously, an application cannot continue execution until the routine is completed.

synthesizer: See free-form, four-tone, or square-wave synthesizer.

synthesizer buffer: A description of the sound to be generated by a synthesizer.

system error alert: An alert box displayed by the System Error Handler.

system error alert table: A resource that determines the appearance and function of system error alerts.

System Error Handler: The part of the Operating System that assumes control when a fatal system error occurs.

system error ID: An ID number that appears in a system error alert to identify the error.

system event mask: A global event mask that controls which types of events get posted into the event queue.

system font: The font that the system uses (in menus, for example). Its name is Chicago.

system font size: The size of text drawn by the system in the system font; 12 points.
system heap: The portion of the heap reserved for use by the Operating System.

system heap zone: The heap zone provided by the Memory Manager for use by the Operating System; equivalent to the system heap.

system resource: A resource in the system resource file.

system resource file: A resource file containing standard resources, accessed if a requested resource wasn't found in any of the other resource files that were searched.

system startup information: Certain configurable system parameters that are stored in the first two logical blocks of a volume and read in at system startup.

system window: A window in which a desk accessory is displayed.

TextEdit: The part of the Toolbox that supports the basic text entry and editing capabilities of a standard Macintosh application.

TextEdit scrap: The place where certain TextEdit routines store the characters most recently cut or copied from text.

thousands separator: The character that separates every three digits to the left of the decimal point.

thumb: The Control Manager's term for the scroll box (the indicator of a scroll bar).

tick: A sixtieth of a second.

Toolbox: Same as User Interface Toolbox.

Toolbox Event Manager: The part of the Toolbox that allows your application program to monitor the user's actions with the mouse, keyboard, and keypad.

Toolbox Utilities: The part of the Toolbox that performs generally useful operations such as fixed-point arithmetic, string manipulation, and logical operations on bits.

track: Disk space composed of 8 to 12 consecutive sectors. A track corresponds to one ring of constant radius around the disk.

transaction: A request-response communication between two ATP clients. See transaction request and transaction response.

transaction ID: An identifier assigned to a transaction.

transaction request: The initial part of a transaction in which one socket client asks another to perform an operation and return a response.

transaction response: The concluding part of a transaction in which one socket client returns requested information or simply confirms that a requested operation was performed.

Transcendental Functions Package: A Macintosh package that contains trigonometric, logarithmic, exponential, and financial functions, as well as a random number generator.
transfer mode: A specification of which Boolean operation QuickDraw should perform when
drawing or when transferring a bit image from one bit map to another.

trap dispatch table: A table in RAM containing the addresses of all Toolbox and Operating
System routines in encoded form.

trap dispatcher: The part of the Operating System that examines a trap word to determine what
operation it stands for, looks up the address of the corresponding routine in the trap dispatch
table, and jumps to the routine.

trap macro: A macro that assembles into a trap word, used for calling a Toolbox or Operating
System routine from assembly language.

trap number: The identifying number of a Toolbox or Operating System routine; an index into
the trap dispatch table.

trap word: An unimplemented instruction representing a call to a Toolbox or Operating System
routine.

unimplemented instruction: An instruction word that doesn't correspond to any valid
machine-language instruction but instead causes a trap.

unit number: The number of each device driver's entry in the unit table.

unit table: A 128-byte nonrelocatable block containing a handle to the device control entry for
each device driver.

unlock: To allow a relocatable block to be moved during heap compaction.

unmounted volume: A volume that hasn't been inserted into a disk drive and had descriptive
information read from it, or a volume that previously was mounted and has since had the memory
used by it released.

unpurgeable block: A relocatable block that can't be purged from the heap.

update event: An event generated by the Window Manager when a window's contents need to
be redrawn.

update region: A window region consisting of all areas of the content region that have to be
redrawn.

user bytes: Four bytes in an ATP header provided for use by ATP's clients.

User Interface Toolbox: The software in the Macintosh ROM that helps you implement the
standard Macintosh user interface in your application.

validity status: A number stored in parameter RAM designating whether the last attempt to
write there was successful. (The number is $A8 if so.)

variation code: The part of a window or control definition ID that distinguishes closely related
types of windows or controls.
VBL task: A task performed during the vertical retrace interrupt.

vector table: A table of interrupt vectors in low memory.

version data: In an application's resource file, a resource that has the application's signature as its resource type; typically a string that gives the name, version number, and date of the application.

version number: A number from 0 to 255 used to distinguish between files with the same name.

Versatile Interface Adapter (VIA): The chip that handles most of the Macintosh's I/O and interrupts.

vertical blanking interrupt: See vertical retrace interrupt.

vertical blanking interval: The time between the display of the last pixel on the bottom line of the screen and the first one on the top line.

vertical retrace interrupt: An interrupt generated 60 times a second by the Macintosh video circuitry while the beam of the display tube returns from the bottom of the screen to the top; also known as vertical blanking interrupt.

Vertical Retrace Manager: The part of the Operating System that schedules and executes tasks during the vertical retrace interrupt.

vertical retrace queue: A list of the tasks to be executed during the vertical retrace interrupt.

VIA: See Versatile Interface Adapter.

view rectangle: In TextEdit, the rectangle in which the text is visible.

visible control: A control that's drawn in its window (but may be completely overlapped by another window or other object on the screen).

visible window: A window that's drawn in its plane on the desktop (but may be completely overlapped by another window or object on the screen).

visRgn: The region of a grafPort, manipulated by the Window Manager, that's actually visible on the screen.

volume: A piece of storage medium formatted to contain files; usually a disk or part of a disk. A 400K-byte 3 1/2-inch Macintosh disk is one volume.

volume allocation block map: A list of 12-bit entries, one for each allocation block, that indicate whether the block is currently allocated to a file, whether it's free for use, or which block is next in the file. Block maps exist both on volumes and in memory.

volume attributes: Information contained on volumes and in memory indicating whether the volume is locked, whether it's busy (in memory only), and whether the volume control block matches the volume information (in memory only).
volume buffer: Memory used initially to load the master directory block, and used thereafter for reading from files that are opened without an access path buffer.

volume control block: A nonrelocatable block that contains volume-specific information, including the volume information from the master directory block.

volume-control-block queue: A list of the volume control blocks for all mounted volumes.

volume index: A number identifying a mounted volume listed in the volume-control-block queue. The first volume in the queue has an index of 1, and so on.

volume information: Volume-specific information contained on a volume, including the volume name and the number of files on the volume.

volume name: A sequence of up to 27 printing characters that identifies a volume; followed by a colon (:) in File Manager routine calls, to distinguish it from a file name.

volume reference number: A unique number assigned to a volume as it's mounted, used to refer to the volume.

waveform: The physical shape of a wave.

waveform description: A sequence of bytes describing a waveform.

wavelength: The horizontal extent of one complete cycle of a wave.

window: An object on the desktop that presents information, such as a document or a message.

window class: In a window record, an indication of whether a window is a system window, a dialog or alert window, or a window created directly by the application.

window definition function: A function called by the Window Manager when it needs to perform certain type-dependent operations on a particular type of window, such as drawing the window frame.

window definition ID: A number passed to window-creation routines to indicate the type of window. It consists of the window definition function's resource ID and a variation code.

window frame: The structure region of a window minus its content region.

window list: A list of all windows ordered according to their front-to-back positions on the desktop.

Window Manager: The part of the Toolbox that provides routines for creating and manipulating windows.

Window Manager port: A graffPort that has the entire screen as its portRect and is used by the Window Manager to draw window frames.

window record: The internal representation of a window, where the Window Manager stores all the information it needs for its operations on that window.
window template: A resource that contains information from which the Window Manager can create a window.

word: In TextEdit, any series of printing characters, excluding spaces (ASCII code $20) but including nonbreaking spaces (ASCII code $CA).

word wraparound: Keeping words from being split between lines when text is drawn.

write data structure: A data structure used to pass information to the ALAP or DDP modules.

zone: An arbitrary subset of AppleTalk networks in an internet. See also heap zone.

zone header: The internal "housekeeping" information maintained by the Memory Manager at the beginning of each heap zone.

zone pointer: A pointer to a zone record.

zone record: A data structure representing a heap zone.

zone trailer: A minimum-size free block marking the end of a heap zone.
INDEX

A

ABByte data type II-276
ABCaltType data type II-274
ABProtoType data type II-274
ABRecHandle data type II-274
ABRecPtr data type II-274
ABusRecord data type II-274
ALAP parameters II-276
ATP parameters II-287
DDP parameters II-281
NBP parameters II-298
ABusVars global variable II-328
access path II-83
access path buffer II-84
ACount global variable I-423
action procedure I-316, 324, 328
in control definition function I-332
activate event I-244, 279
event message I-252
active
control I-313
window I-46, 270, 284
AddPt procedure I-193
AddrBlock data type II-281
AddResMenu procedure I-353
AddResource procedure I-124
AddResponse function II-318
address mark II-211
ALAP See AppleTalk Link Access Protocol
ALAP frame II-264
ALAP protocol type II-264
alert I-401, 409
guidelines I-68
alert box I-401
Alert function I-418
alert stages I-409
alert template I-403, 424
resource format I-426
alert window I-402
AlertTemplate data type I-424
AlertTHndl data type I-425
AlertTPtr data type I-425
alias II-266
Allocate function
high-level II-94
low-level II-113
allocated block II-10
allocation block II-79
amplitude of a wave II-223
AngleFromSlope function I-476
ANumber global variable I-423
ApFontID global variable I-219
AppendMenu procedure I-352
AppFile data type II-58
Apple menu I-54
AppleTalk address II-265
AppleTalk Link Access Protocol II-263
assembly language II-306
data reception II-325
Pascal II-276
AppleTalk Manager I-13; II-261, 271
assembly language II-304
Pascal II-273
AppleTalk Transaction Protocol II-266, 267
assembly language II-312
Pascal II-287
application font I-219
application heap I-74; II-9
limit II-17, 29
application parameters II-20
application space II-20
application window I-270
ApplcZone function II-32
ApplLimit global variable II-19, 21, 29
ApplScratch global variable I-85
ApplZone global variable II-19, 21, 32
AppParmHandle global variable II-57
arrow cursor I-163, 167
arrow global variable I-147, 163
ascent of a font I-228
in TextEdit I-378
ASCII codes I-247
assembly language I-83
asynchronous communication II-245
asynchronous execution
AppleTalk Manager II-273
Device Manager II-180
File Manager II-97
at-least-once transaction II-266
ATP See AppleTalk Transaction Protocol
ATPAddRsp function II-295
ATPCloseSocket function II-291
ATPGetRequest function II-293
ATPLoad function II-290
ATPOpenSocket function II-290
ATPReqCancel function II-293
ATPRequest function II-292
ATPResponse function II-296
ATPRspCancel function II-296
ATPSndRequest function II-291
ATPSndRsp function II-294
ATPUndload function II-290
AttachPH function II-308
auto-key event 1-244, 246
auto-key rate 1-246; II-371
auto-key threshold 1-246; II-371
auto-pop bit I-89
automatic scrolling I-48
in TextEdit I-380

BlockMove procedure II-44
boot blocks See system startup information
boundary rectangle I-144
break II-246
bridge II-265
BringToFront procedure I-286
broadcast service II-264
BufPtr global variable II-19, 21
BufTgDate global variable II-212
BufTgFBkNum global variable II-212
BufTgFFlag global variable II-212
BufTgFNum global variable II-212
bundle II-85; III-11
resource format III-12
Button function I-259
button type of control I-311, 404
Byte data type I-78

CalcMenuSize procedure I-361
CalcVBehind procedure I-297
CalcVis procedure I-297
CalcVisBehind procedure I-297
caret I-376, 379
caret-blink time I-260; II-371
CareTTime global variable I-260
CautionAlert function I-420
Chain procedure II-59
ChangedResource procedure I-123
character codes I-246
character device II-175
character image I-227
character keys I-33, 246
character offset I-228
character origin I-228
character position I-375
character rectangle I-228
character set I-247
character style I-151
of menu items I-348, 360
character width I-173, 228
Chars data type I-384
CharsHandle data type I-384
CharsPtr data type I-384
CharWidth function I-173
check box I-312, 404
check mark in a menu I-347, 358
CheckItem procedure I-358
CheckUpdate function I-296
ClearMenuBar procedure I-354
Index

click See mouse-down event
click loop routine I-380
ClipAbove procedure I-296
Clipboard I-58 See also scrap
clipping region of a graffPort I-149
ClipRect procedure I-167
clipRgn of a graffPort I-149
clock chip II-369
  hardware III-36
close box See go-away region
Close command I-56
Close function, high-level
  Device Manager II-178
  File Manager II-94
Close function, low-level
  Device Manager II-184
  File Manager II-114
close routine
  of a desk accessory I-446
  of a driver II-187, 193
CloseATPSkt function II-316
closed device driver II-176
closed file II-83
CloseDeskAcc procedure I-440
CloseDialog procedure I-413
CloseDriver function II-178
ClosePgon procedure I-190
ClosePicture procedure I-189
ClosePoly procedure I-190
ClosePort procedure I-164
CloseResFile procedure I-115
CloseRgn procedure I-182
CloseSkt function II-312
CloseWindow procedure I-283
ClrAppFiles procedure II-58
CmpString function II-377
color drawing I-158, 173
ColorBit procedure I-174
Command-key equivalent See keyboard equivalent
Command-period II-154
Command-Shift-number I-258
commands I-51, 341
compaction, heap I-74; II-12, 39
CompactMem function II-39
completion routine
  Device Manager II-180, 181
  File Manager II-97, 99
  Sound Driver II-231
ConfirmName function I-323
content region of a window I-271
control I-65, 311
  defining your own I-328
  in a dialog/alert I-404
control definition function I-314, 328
control definition ID I-315, 328
Control function
  high-level II-179
  low-level II-186
control information II-176
control list I-274, 317
Control Manager I-11, 309
  routines I-319
control record I-316
control routine
  of a desk accessory I-446
  of a driver II-187, 194
control template I-315
  resource format I-332
ControlHandle data type I-317
ControlPtr data type I-317
ControlRecord data type I-317
coordinate plane I-138
CopyBits procedure I-188
CopyRgn procedure I-183
CouldAlert procedure I-420
CouldDialog procedure I-415
CountAppFiles procedure II-57
CountMitems function I-361
CountResources function I-118
CountTypes function I-117
Create function
  high-level II-90
  low-level II-107
CreateResFile procedure I-114
creator of a file III-9
CrsrThresh global variable II-372
CurActivate global variable I-280
CurApName global variable II-58
CurApRefNum global variable II-58
CurDeactive global variable I-280
CurJTOffset global variable II-62
CurMap global variable I-117
CurPageOption global variable II-60
CurPitch global variable II-226, 232
current heap zone II-10, 31
current resource file I-105, 116
CurrentA5 global variable I-95; II-19, 21, 386
CurResFile function I-116
CursHandle data type I-474

III-263
cursor I-146
QuickDraw routines I-167
standard cursors I-147, 474
utility routines I-474
Cursor data type I-146
cursor level I-167
CursPtr data type I-474
CurStackBase global variable II-19, 21, 358
cut and paste I-59
intelligent I-63
in TextEdit I-385

D
DABeeper global variable I-411
DAStrings global array I-421
data bits II-245
data buffer II-83, 176
data fork I-105; II-81
data mark II-211
datagram II-265
loss recovery II-268
Datagram Delivery Protocol II-265
assembly language II-308
Pascal II-281
date operations II-377
Date2Secs procedure II-379
DateForm data type I-504
date/time record II-377
DateTimeRec data type II-378
DCtlEntry data type II-190
DCtlHandle data type II-190
DCtlPtr data type II-190
DDP See Datagram Delivery Protocol
DDPCloseSocket function II-282
DDPOpenSocket function II-282
DDPRdCancel function II-284
DDPRead function II-283
DDPWrite function II-283
default button
  in an alert I-69, 401, 424
  in a dialog I-67, 400, 407
default volume II-80
  getting See GetVol function
  setting See SetVol function
DefltStack global variable II-17
DefVCBPtr global variable II-126
Delay procedure II-384
Delete function
  high-level II-97
  low-level II-119

DeleteMenu procedure I-354
DeltaPoint function I-475
Dequeue function II-383
dereferencing a handle II-14
descent of a font I-228
desk accessory I-437
  writing your own I-443
Desk Manager I-12, 435
  routines I-440
desk scrap I-453
  data types I-454
  format I-462
  routines I-457
DeskHook global variable I-282, 288
DeskPattern global variable I-282
desktop I-32, 269
Desktop file III-10
destination rectangle I-374
DetachPH function II-308
DetachResource procedure I-120
device II-175
device control entry II-189
device driver I-13; II-175
  for a desk accessory I-443
  structure II-187
  writing your own II-193
device driver event I-244
Device Manager I-13; II-173
Device Manager routines II-177
  device control entry access II-190
  high-level II-178
  low-level II-180
  for writing drivers II-194
dial I-312
dialog box I-66, 399
Dialog Manager I-12, 397
  routines I-411
dialog pointer I-407
dialog record I-403, 407
dialog template I-402, 403
  resource format I-425
dialog window I-401
DialogPeek data type I-408
DialogPtr data type I-407
DialogRecord data type I-408
DialogSelect function I-417
DialogTemplate data type I-423
DialogTHndl data type I-424
DialogTPtr data type I-424
DIBadMount function II-396
DiffRgn function I-398
DIFormat function II-398

III-264
Index

DragControl procedure I-325
DragGrayRgn function I-294
DragHook global variable
Control Manager I-324, 326
Window Manager I-288, 289, 290, 295
DragPattern global variable
Control Manager I-324, 326
Window Manager I-295
DragTheRgn function I-295
DragWindow procedure I-289
DrawChar procedure I-172
DrawControls procedure I-322
DrawDialog procedure I-418
DrawGrowIcon procedure I-287
drawing I-155
color I-158, 173
DrawMenuBar procedure I-354
DrawNew procedure I-296
DrawPicture procedure I-190
DrawString procedure I-172
DrawText procedure I-172
drive number II-80
drive queue II-127
driver See device driver
driver I/O queue II-180, 191
driver name II-176
driver reference number II-176
DriveStatus function II-215
DrvQEi data type II-127
DrvQHdr global variable II-128
DrvSts data type II-215
DSAlertRect global variable II-362
DSAlertTab global variable II-359, 362
DSErrCode global variable II-362

E

Edit menu I-58
and desk accessories I-441, 447
edit record I-374
Eject function
high-level II-90
low-level II-107
Elems68K See Transcendental Functions
Package
empty handle I-76; II-14, 40
EmptyHandle procedure II-40
EmptyRect function I-176
EmptyRgn function I-186
enabled
dialog/alert item 1-405
menu 1-358
menu item I-358
EnableItem procedure 1-358
end-of-file II-81
end-of-message flag II-270
EndUpdate procedure 1-293
Enqueue procedure II-382
entity name II-265, 298
EntityName data type II-298
Environ procedure II-385
EntityPtr data type II-298
equal-tempered scale II-237
EqualPt function I-193
EqualRect function I-176
EqualRgn function I-185
EqualString function II-377
EraseArc procedure I-180
EraseOval procedure I-178
ErasePoly procedure I-192
EraseRect procedure I-177
EraseRgn procedure I-186
EraseRoundRect procedure I-179
error number See result code
ErrorSound procedure I-411
event I-243
  priority I-245
event code I-249
Event Manager, Operating System I-13; II-65
  routines II-68
Event Manager, Toolbox I-11, 241
  routines I-257
  event mask I-253
  event message I-249
  event queue I-243
  structure II-70
  event record I-249
  event types I-244
EventAvail function I-259
EventQueue global variable II-71
EventRecord data type I-249
EvQEi data type II-71
exactly-once transaction II-266
example program I-13
exception II-195
exception vector III-17
ExitToShell procedure II-59
exponential functions II-407
extended selection I-39
  in TextEdit I-384
external file system II-128
  external reference I-95
  ExtStsDT global variable II-199

F

FCBSPtr global variable II-127
Fetch function II-194
FFSynthPtr data type II-228
FFSynthRec data type II-228
file II-79, 81
file control block II-126
file-control-block buffer II-126
file creator III-9
file directory II-79, 122
file icon II-85; III-10
file I/O queue II-97, 124
File Manager I-13; II-77
File Manager routines
  high-level II-88
  low-level II-97
  for queue access II-125, 126, 128
File menu I-55
file name II-81
file number II-122
file reference III-10
  resource format III-12
file tags II-212
file tags buffer II-212
file type III-9
fileFilter function I-524
FillArc procedure I-181
FillOval procedure I-178
FillPoly procedure I-192
FillRect procedure I-177
FillRgn procedure I-187
FillRoundRect procedure I-179
filterProc function I-415
financial functions II-407
FindControl function I-323
Finder information II-55
Finder interface II-55, 84; III-7
FinderName global variable II-59
FindWindow function I-287
FInfo data type II-84
FInitQueue procedure II-103
Fixed data type I-79
fixed-point
  arithmetic I-467
  numbers I-79
fixed-width font I-228
FixMul function I-467
FixRatio function  I-467
FixRound function  I-467
FlashMenuBar procedure  I-361
Floating-Point Arithmetic Package  I-13; II-403
FlushEvents procedure  II-69
FlushFile function  II-114
FlushVol function
  high-level  II-89
  low-level  II-105
FMInput data type  I-224
FMOOutPtr data type  I-227
FMOOut data type  I-227
FMSwapFont function  I-223
folder  II-85
font  I-60, 151, 217
  characters  I-220
  format  I-227
  resource format  I-234
  resource ID  I-234
font characterization table  I-225
font height  I-228
Font Manager  I-11, 215
  communication with QuickDraw  I-224
  routines  I-222
Font menu  I-60, 353
font number  I-217, 219
font record  I-230
font rectangle  I-228
font scaling  I-220
font size  I-153, 217
FontInfo data type  I-173
FontRec data type  I-231
FontSize menu  I-61
ForeColor procedure  I-173
font  I-105; II-81
four-tone record  II-227
four-tone synthesizer  II-223, 226
FP68K  See Floating-Point Arithmetic Package
frame
  ALAP  II-264
  picture  I-158
  serial communication  II-246
  stack  I-96; II-17
  window  I-271
frame check sequence  II-265
frame header  II-264
frame pointer (stack)  I-96
frame trailer  II-264
FrameArc procedure  I-180
FrameOval procedure  I-177
FramePoly procedure  I-192
FrameRect procedure  I-176
FrameRgn procedure  I-186
FrameRoundRect procedure  I-178
framing error  II-246
free-form synthesizer  II-223, 228
free memory block  II-10
FreeAlert procedure  I-420
FreeDialog procedure  I-415
FreeMem function  II-38
FreeWave data type  II-228
frequency of a wave  II-223
FrontWindow function  I-286
FScaleDisable global variable  I-222
FSClose function  II-94
FSDelete function  II-97
FSOpen function  II-91
FSQHdr global variable  II-125
FSRead function
  Device Manager  II-178
  File Manager  II-92
FSWrite function
  Device Manager  II-179
  File Manager  II-92
FTSndRecPtr data type  II-227
FTSoundRec data type  II-227
FTSynthPtr data type  II-227
FTSynthRec data type  II-227
full-duplex communication  II-245

G
GetAlrtStage function  I-422
GetAppFiles procedure  II-58
GetApplLimit function  II-29
GetAppParms procedure  II-58
GetCareTime function  I-260
GetClip procedure  I-167
GetCRefCon function  I-327
GetCTitle procedure  I-321
GetCtlAction function  I-328
GetCtlMax function  I-327
GetCtlMin function  I-327
GetCtlValue function  I-326
GetCursor function  I-474
GetDateTime procedure  II-378
GetDb1Time function  I-260
GetDCtlEntry function  II-190
GetDIteItem procedure  I-421
GetDrvQHdr function  II-128
GetEOF function
  high-level  II-93
  low-level  II-112
GetEvQHdr function  II-71
GetFilelnfo function
  high-level  II-95
  low-level  II-115
GetFInfo function  II-95
GetFName procedure  I-223
GetFNum procedure  I-223
GetFontlnfo procedure  I-173
GetFontName procedure  I-223
GetFPos function
  high-level  II-92
  low-level  II-111
GetFSQHdr function  II-125
GetHandleSize function  II-33
GetIcon function  I-473
GetIndPattern procedure  I-473
GetIndResource function  I-118
GetIndString procedure  I-468
GetIndType procedure  I-117
GetItem procedure  I-358
GetItemIcon procedure  I-360
GetItemMark procedure  I-359
GetItemStyle procedure  I-360
GetText procedure  I-422
GetHTMLIcon procedure  I-360
GetHlmMark procedure  I-359
GetHlmStyle procedure  I-360
GetKeys procedure  I-259
GetMaxCtl function  I-327
GetMenu function  I-351
GetMenuBar function  I-355
GetMHandle function  I-361
GetMinCtl function  I-327
GetMouse procedure  I-259
GetNamedResource function  I-119
GetNewControl function  I-321
GetNewDialog function  I-413
GetNewMBar function  I-354
GetNewWindow function  I-283
GetNextEvent function  I-257
GetNodeAddress function  II-303
GetOSEvent function  II-69
GetPattern function  I-473
GetPen procedure  I-169
GetPenState procedure  I-169
GetPicture function  I-475
GetPixel function  I-195
GetPort procedure  I-165
GetPtrSize function  II-37
GetRequest function  II-317
GetResAttrs function  I-121
GetResFileAttrs function  I-127
GetResInfo procedure  I-121
GetResource function  I-119
GetRMenu function  I-351
GetScrap function  I-469
GetSoundVol procedure  II-232
GetString function  I-468
GetSysPPtr function  II-381
GetTime procedure  II-380
GetTrapAddress function  II-384
GetVBLQHdr function  II-352
GetVCBQHdr function  II-126
GetVlnfo function  II-89
GetVol function
  high-level  II-89
  low-level  II-104
GetVolnfo function
  high-level  II-89
  low-level  II-104
GetVRefNum function  II-89
GetWindowPic function  I-293
GetWMgrPort procedure  I-282
GetWRefCon function  I-293
GetWTitle procedure  I-284
GetZone function  II-31
GhostWindow global variable  I-287
global coordinates  I-155
global variables
  list  III-227
QuickDraw  I-138, 162
GlobalToLocal procedure  I-193
go-away region of a window  I-271, 288
GrafDevice procedure  I-165
grafPort  I-147
  routines  I-162
GrafPort data type  I-148
GrafPtr data type  I-148
GrafVerb data type  I-198
gray global variable  I-162
GrayRgn global variable  I-282, 296
grow image of a window  I-289
grow region of a window  I-272, 289
grow zone function  II-14, 42
GrowWindow function  I-289
GZRootHnd global variable  II-43
GZSaveHnd function  II-43

H
HandAndHand function  II-375
handle I-75, 78; II-12
dereferencing II-14
empty II-40
manipulation II-374
Handle data type I-78
HandleZone function II-34
HandToHand function II-374
hardware III-15
hardware overrun error II-246
heap I-12, 23; II-9, 17
compaction I-74; II-12, 39
creating on the stack II-45
zone II-9, 22
HeapEnd global variable II-19, 21
HideControl procedure I-322
HideCursor procedure I-168
HidePen procedure I-168
HideWindow procedure I-283
highlighted I-31
control I-313
menu title I-357
window I-270
HiliteControl procedure I-322
HiliteMenu procedure I-357
HiliteWindow procedure I-286
HiWord function I-472
HLock procedure II-41
HNoPurge procedure II-42
HomeResFile function I-117
horizontal blanking interval III-18
hotSpot of a cursor I-146
HPurge procedure II-41
HUnlock procedure II-41

I

icon I-32
in a dialog/alert I-404
for a file II-85; III-10
in a menu I-347, 359
utility routines I-473
icon list III-11
resource format I-476; III-12
icon number I-347
image width I-228
inactive
control I-313
window I-46, 270
indicator of a dial I-312
InfoScrap function I-457
InitAllPacks procedure I-484
InitApplZone procedure II-28
InitCursor procedure I-167
InitDialogs procedure I-411
InitFonts procedure I-222
InitGraf procedure I-162
InitMenus procedure I-351
InitPack procedure I-484
InitPort procedure I-164
InitQueue procedure II-103
InitResources function I-114
InitUtil function II-380
InitWindows procedure I-281
InitZone procedure II-29
input driver II-246
insertion point I-41, 375
InsertMenu procedure I-353
InsertResMenu procedure I-353
InsetRect procedure I-175
InsetRgn procedure I-184
Int64Bit data type I-472
interface routine I-95
international resources I-495
International Utilities Package I-12, 493
routines I-504
internet II-265
internet address II-265, 314
interrupt II-195
level-1 (VIA) II-197; III-38
level-2 (SCC) II-198
level-3 II-196
vertical retrace II-349
interrupt handler II-195
writing your own II-200
interrupt priority level II-196
interrupt vector II-196
Intl0Hndl data type I-496
Intl0Ptr data type I-496
Intl0Rec data type I-497
Intl1Hndl data type I-500
Intl1Ptr data type I-500
Intl1Rec data type I-500
InvalRect procedure I-291
InvalRgn procedure I-291
InverRect procedure I-177
InverRgn procedure I-186
InverRoundRect procedure I-179
InvertArc procedure I-181
InvertOval procedure I-178
InvertPoly procedure I-192
InvertRect procedure I-177
InvertRgn procedure I-186
InvertRoundRect procedure I-179

III-269
invisible
  control I-316
  dialog/alert item I-406
  file icon II-85
  window I-274
IODone function II-195
I/O queue See driver I/O queue or file I/O queue
I/O request II-97, 180
IsATPOpen function II-304
IsDialogEvent function I-416
IsMPPOpen function II-304
item
dialog/alert I-403
  menu I-341
item list I-403
  resource format I-427
item number
dialog/alert I-406
  menu I-350
item type I-404
IUC ompString function I-506
IUDatePString procedure I-505
IUDateString procedure I-504
IUEqualString function I-506
IUGetInt function I-505
IUMagIDString function I-507
IUMagString function I-506
IUMetric function I-505
IUS etInt procedure I-506
IUTimePString procedure I-505
IUT imeString procedure I-505
IWM III-17
IWM global variable III-34

J

J Fetch global variable II-194
J IODone global variable II-195
job dialog I-149
job subrecord II-150
journal code I-262
JournalFlag global variable I-261
journaling mechanism I-261
JournalRef global variable I-261
JStash global variable II-195
jump table I-60
jump vector II-194
just-tempered scale II-237
justification I-376
  setting I-387

K

kerning I-152, 228
key codes I-250
key-down event I-244
  responding to I-256
key-up event I-244, 254
keyboard I-33
  hardware III-29
keyboard configuration I-248
keyboard equivalent I-343
  meta-character I-348
  responding to I-356
  standard equivalents I-53
keyboard event I-244, 246
  event message I-250
  responding to I-256
keyboard touch See auto-key threshold
KeyMap data type I-260
keypad I-35
  hardware III-29
KeyRepThresh global variable I-246
KeyThresh global variable I-246
KillControls procedure I-321
KillIO function
  high-level II-179
  low-level II-187
KillPicture procedure I-190
KillPoly procedure I-191

L

LAPAdrBlock data type II-276
LAPCloseProtocol function II-277
LAPOpenProtocol function II-277
LAPRdCancel function II-279
LAPRead function II-278
LAPWrite function II-277
Launch procedure II-60
leading I-228
ligatures I-501
line height I-378
Line procedure I-171
LineTo procedure I-170
list separator I-497
Lo3Bytes global variable I-85; II-25
LoadNBP function II-324
LoadResource procedure I-119
LoadScrap function I-458
LoadSeg procedure II-60
local coordinates I-153
local ID III-10
LocalToGlobal procedure I-193
location table I-231
lock bit II-25
locked block I-76; II-10
locked file II-84
locked volume II-80
locking a block I-76; II-41
LodeScrap function I-458
logarithmic functions II-407
logical block II-119
logical end-of-file II-81
logical operations I-471
logical size of a block II-22
LongMul procedure I-472
LookupName function II-323
LoWord function I-472
ltGray global variable I-162
Lvl1DT global variable II-197
Lvl2DT global variable II-198

M

magnitude of a wave II-223
main event loop I-16
main segment II-55
MapPoly procedure I-197
MapPt procedure I-196
MapRect procedure I-196
MapRgn procedure I-196
mark
  in a file II-82
  in a menu I-347, 359
mark state II-245
master directory block II-120
master pointer I-75; II-12
  allocation II-22, 31
  structure II-25
MaxAppZone procedure II-30
MaxMem function II-38
MBarEnable global variable I-356, 446
MBarHook global variable I-356
MemError function II-44
memory block I-73; II-10
memory management II-7
  introduction I-71
Memory Manager I-12; II-7
  routines II-27
memory organization II-19
MemTop global variable II-19, 21, 44

menu I-341
  defining your own I-362
  guidelines I-51
  resource format I-364
  standard menus I-54, 342
menu bar I-341
  resource format I-365
menu definition procedure I-344, 362
menu ID I-344
menu item I-341
  blinking I-361; II-371
  menu item number I-350
menu list I-345
Menu Manager I-12, 339
  routines I-351
  menu record I-344
  menu title I-341
MenuFlash global variable I-361
MenuHandle data type I-345
MenuHook global variable I-356
MenuInfo data type I-345
MenuKey function I-356
MenuList global variable I-346
MenuPtr data type I-345
MenuSelect function I-355
meta-characters
  AppleTalk Manager II-266, 320
  Menu Manager I-346
  MinStack global variable II-17
  MinusOne global variable I-85
  missing symbol I-152, 220, 230
  modal dialog box I-67, 400, 415
  ModalDialog procedure I-415
  modeless dialog box I-67, 400, 416
  modes I-28
  modifier flags I-252
  modifier keys I-34, 246
    flags in event record I-252
  MoreMasters procedure II-31
  mounted volume II-79
  MountVol function II-103
mouse I-36
  hardware III-25
  mouse-down event I-244
    responding to I-255
  mouse scaling II-372
  mouse-scaling threshold II-372
  mouse-up event I-244
    responding to I-255
Move procedure I-170
MoveControl procedure I-325

III-271
MoveHHi procedure II-44
MovePortTo procedure I-166
MoveTo procedure I-170
MoveWindow procedure I-289
MPP II-271
MPPClose function II-275
MPPOpen function II-275
Munger function I-468

N

Name-Binding Protocol II-266
assembly language II-319
Pascal II-298
name lookup II-266
names directory II-266
names information socket II-266
names table II-266, 321
NBP See Name-Binding Protocol
NBP tuple II-266
NBPConfirm function II-301
NBPExtract function II-300
NBPLoad function II-301
NBPLookup function II-300
NBPRегист function II-299
NBPUnregister function II-301
network event I-244; II-275
network number II-265
network-visible entity II-265
New command I-56
NewControl function I-319
NewDialog function I-412
NewHandle function I-76, 80; II-32
newline character II-84
newline mode II-84
NewMenu function I-351
NewPtr function I-75, 79; II-36
NewRgn function I-181
NewString function I-468
NewWindow function I-282
node II-263
node ID II-263
nonbreaking space II-246
nonrelocatable block I-75; II-10
  allocating II-36
  releasing II-36
NoteAlert function I-420
null event I-245
NumToString procedure I-489

O

ObscureCursor procedure I-168
off-line volume II-80
OffLine function II-106
OffsetPoly procedure I-191
OffsetRect procedure I-174
OffsetRgn function I-183
offset/width table II-231
OffsetRgn procedure I-183
OldContent global variable I-296
OldStructure global variable I-296
on-line volume II-80
OneOne global variable I-85
Open command I-56
open device driver II-176
open file II-83
Open function, high-level
  Device Manager II-178
  File Manager II-91
Open function, low-level
  Device Manager II-184
  File Manager II-108
open permission II-83
open routine
  of a desk accessory I-445
  of a driver II-187, 193
OpenATPSkt function II-315
OpenDeskAcc function I-440
OpenDriver function II-178
OpenPicture function I-189
OpenPoly function I-190
OpenPort procedure I-163
OpenResFile function I-115
OpenRF function
  high-level II-91
  low-level II-109
OpenRgn procedure I-181
OpenSkt function II-311
Operating System I-9
  queues II-372
Operating System Event Manager I-13; II-65
  routines II-68
Operating System Utilities I-13; II-367
  routines II-374
OSErr data type II-373
OSEventAvail function II-70
OSType data type II-373
output driver II-246
overrun error See hardware overrun error or
  software overrun error
owned resources I-109
Pack2 See Disk Initialization Package
Pack3 See Standard File Package
Pack4 See Floating-Point Arithmetic Package
Pack5 See Transcendental Functions Package
Pack6 See International Utilities Package
Pack7 See Binary-Decimal Conversion Package
Package Manager I-12, 481
packages I-12, 483
PackBits procedure I-470
page rectangle II-150
Page Setup command I-57
PaintArc procedure I-180
PaintBehind procedure I-297
PaintOne procedure I-296
PaintOval procedure I-178
PaintPoly procedure I-192
PaintRect procedure I-177
PaintRgn procedure I-186
PaintRoundRect procedure I-179
PaintWhite global variable I-297
palette I-32
pane I-49
panel I-50
paper rectangle II-150
ParamBlkType data type II-98, 181
ParamBlockRec data type II-98, 181
driver I/O queue entry II-191
file I/O queue entry II-124
parameter block I-93; II-97, 180
parameter RAM II-369
default values II-370
routines II-380
ParamText procedure I-421
parity bit II-245
parity error II-246
ParmBlkPtr data type II-98, 181
part code I-315, 330
path reference number II-83
PatHandle data type I-473
PatPtr data type I-473
pattern I-145, 473
Pattern data type I-146
pattern list I-473
resource format I-476
pattern transfer mode I-157
PBAlocate function II-113
PBClose function
Device Manager II-184
File Manager II-114
PBControl function II-186
PBCreate function II-107
PBDel ete function II-119
PBEject function II-107
PBFflushFile function II-114
PBFflushVol function II-105
PBGetEOF function II-112
PBGetFInfo function II-115
PBGetFPos function II-111
PBGetVInfo function II-104
PBGetVol function II-104
PBKillIO function II-187
PBMountVol function II-103
PBOffLine function II-106
PBOpen function
Device Manager II-185
File Manager II-108
PBOpenRF function II-109
PBRead function
Device Manager II-185
File Manager II-110
PBRename function II-118
PBResetLock function II-117
PBSetEOF function II-112
PBSetFInfo function II-116
PBSetFLock function II-116
PBSetFPos function II-111
PBSetFVers function II-117
PBSetVol function II-105
PBSstatus function II-186
PBUnmountVol function II-106
PBWrite function
Device Manager II-185
File Manager II-110
pen characteristics I-150
PenMode procedure I-169
PenNormal procedure I-170
PenPat procedure I-170
PenSize procedure I-169
PenState data type I-169
period of a wave II-223
phase of a wave cycle II-223
physical end-of-file II-81
physical size of a block II-23
PicComment procedure I-189
PicHandle data type I-159
PicPtr data type I-159
picture I-158
QuickDraw routines I-189
utility routine I-475
picture comments I-159
Picture data type I-159
picture frame I-158
PinRect function I-293
pixel I-139, 143
PlotIcon procedure I-473
point (coordinate plane) I-139
routines I-193
point (font size) I-61, 153, 217
Point data type I-139
pointer (to memory) I-75, 78
manipulation II-374
type coercion I-79
pointer (on screen) I-36, 37 See also cursor
polygon I-159
Point data type I-139
PolyHandle data type I-160
PolyPtr data type I-160
portBits of a grafPort I-148
PortBUse global variable II-305
portRect of a grafPort I-149
PortSize procedure I-165
post an event I-243
PostEvent function II-68
PrClose procedure II-157
PrCloseDoc procedure II-160
PrClosePage procedure II-160
PrCtlCall procedure II-163
PrDrvClose procedure II-163
PrDrvDCE function II-163
PrDrvOpen procedure II-163
PrDrvVers function II-163
Pragma function II-161
prime routine of a driver II-187, 193
Print command I-57
print dialogs II-148
print record II-148
PrintDefault procedure II-158
Printer Driver I-13; II-147, 162
printer information subrecord II-150
printer resource file II-147
PrintErr global variable II-161
printing grafPort II-147
Printing Manager I-13; II-145
routines II-157
printing methods II-153
low-level II-164
private scraps I-461
PrJobDialog function II-158
PrJobMerge procedure II-159
processor priority II-196
ProcPtr data type I-78
PrOpen procedure II-157
PrOpenDoc function II-159
PrOpenPage function II-159
proportional font procedure II-159
protocol II-263
protocol handler II-264
writing your own II-324, 326
protocol handler table II-264
PrPicFile procedure II-160
PrSetError procedure II-161
PrStdDialog function II-158
PrValidate function II-158
PScrapStuff data type I-457
Pt2Rect procedure I-175
PtInRect function I-175
PtInRgn function I-185
Ptr data type I-78
PtrAndHand function II-376
PtrToHand function II-375
PtrToXHand function II-375
PtrZone function II-38
PtToAngle procedure I-175
purge bit II-25
purge warning procedure II-23
purgeable block I-76; II-10, 41
PurgeMem procedure II-40
purging a block I-76; II-14, 40
PutScrap function I-459
Q
QDProcs data type I-197
QDProcsPtr data type I-197
QElem data type II-373
QElemPtr data type II-373
QHdr data type II-372
QHdrPtr data type II-373
QTypes data type II-373
queue II-373
drive II-127
driver I/O II-180, 191
file I/O II-97, 124
manipulation II-382
vertical retrace II-350, 352
volume-control-block II-125
QuickDraw I-11, 135
communication with Font Manager I-224
routines I-162
Quit command I-57
Index

radio button I-312, 404
RAM III-17
RAM Serial Driver I-13; II-246
advanced Control calls II-254
Device Manager calls II-248
routines II-249
RAMBase global variable I-87
RAMSDClose procedure II-250
RAMSDOpen function II-249
Random function I-194
random number generator I-194; II-407
randSeed global variable I-163, 194
Read function, high-level
Device Manager II-178
File Manager II-92
Read function, low-level
Device Manager II-185
File Manager II-110
ReadDateTime function II-378
ReadPacket function II-327
ReadRes function II-327
read/write permission II-83
RealFont function I-223
reallocating a block I-76; II-14
ReallocHandle procedure II-35
RecoverHandle function II-35
Rect data type I-141
rectangle I-140
routines I-174
RectInRgn function I-185
RectRgn procedure I-183
reference number of a resource file I-105
reference value
control I-316
window I-274
region I-141
routines I-181
Region data type I-142
register-based routines I-90, 93
register-saving conventions I-94
RegisterName function II-322
relative handle II-24
release timer II-270
ReleaseResource procedure I-120
relocatable block I-75; II-10
allocating II-32
releasing II-33
RelRspCB function II-319
RelTCB function II-319
RemoveName function II-324
Rename function
high-level II-96
low-level II-118
ResErr global variable I-116
ResError function I-116
ResErrProc global variable I-116
ResetAlertStage procedure I-423
ResLoad global variable I-118
resource I-103
within a resource I-127
resource attributes I-111
getting I-121
setting I-122
resource data I-106
resource file I-105
attributes I-126
current I-105, 116
format I-128
resource fork I-105; II-81
resource header I-128
resource ID I-108
of fonts I-234
of owned resources I-109
Resource Manager I-9, 101
routines I-113
resource map I-106
resource name I-110
resource reference I-110
format I-130
resource specification I-103, 107
resource type I-103
list I-107
response BDS II-288, 314
ResrvMem procedure II-39
Restart procedure II-385
RestoreA5 procedure II-386
ResType data type I-107
result code I-116; II-27, 374
assembly language I-94
list III-205
resume procedure I-411; II-358
ResumeProc global variable I-411
RetransType data type II-298
retry count II-266
retry interval II-266
Revert to Saved command I-57
RgnHandle data type I-142
RgnPtr data type I-142
RmveResource procedure I-124
RndSeed global variable I-195
ROM III-18
ROM Serial Driver I-13; II-246
Device Manager calls II-248
routines II-250
ROMBase global variable I-87; II-383; III-18
ROMFont0 global variable I-233
routine selector I-483
routing table II-265
Routing Table Maintenance Protocol II-265
row width I-143
RsrcZoneInit procedure I-114
RstFilLock function
  high-level II-96
  low-level II-117
RstFLock function II-96
RTMP II-265
RTMP socket II-265
RTMP stub II-265

S

sample program I-13
SANE II-405
Save As command I-57
Save command I-57
SaveOld procedure I-296
SaveUpdate global variable I-297
SaveVisRgn global variable I-293
ScalePt procedure I-195
scaling factors I-218
SCC II-22
SCC interrupts II-198
SCCRd global variable II-199; III-25
SCCW r global variable II-199; III-25
scrap
  between applications I-453
  in TextEdit I-373, 388
scrap file I-453
Scrap Manager I-12, 451
  routines I-457
ScrapCount global variable I-457
ScrapHandle global variable I-457
ScrapName global variable I-457
ScrapSize global variable I-457
ScrapState global variable I-457
ScrapStuff data type I-457
Scratch8 global variable I-85
Scratch20 global variable I-85
ScrDmpEnb global variable I-258
screen buffer III-18, 19
screenBits global variable I-145, 163
ScreenRes procedure I-473
ScrHRes global variable I-473
ScrBase global variable II-19, 21
scroll bar I-47, 312
updating I-291
ScrollRect procedure I-187
ScrVRes global variable I-473
SdVolume global variable II-232
Secs2Date procedure II-380
sector II-211
SectRect function I-175
SectRgn procedure I-184
segment II-55
Segment Loader I-12; II-53
  routines II-57
selection range I-375
SelectWindow procedure I-234
SelIText procedure I-422
SendBehind procedure I-286
SendRequest function II-316
SendResponse function II-317
sequence number of a datagram II-266
SerClrBrk function II-253
SerGetBuf function II-253
SerHShake function II-251
serial communication II-245
  hardware III-22
Serial Communications Controller III-22
serial data II-245
Serial Drivers I-13; II-243
  advanced Control calls II-254
  Device Manager calls II-248
  routines II-249
SerReset function II-250
SerSetBrk function II-252
SerSetBuf function II-251
SerShk data type II-252
SerStaRec data type II-253
SerStatus function II-253
SetAppBase procedure II-28
SetAppBase procedure II-28
SetApplLimit procedure II-30
SetClikLoop procedure I-390
SetClip procedure I-166
SetCRefCon procedure I-327
SetCTitle procedure I-321
SetCtlAction procedure I-328
SetCtlMax procedure I-327
SetCtlMin procedure I-326
SetCtlValue procedure I-326
SetCursor procedure I-167
SetDAFont procedure I-412
SetDateTime function II-379
SetDItem procedure I-421
SetEmptyRgn procedure I-183
SetEOF function
  high-level II-93
  low-level II-112
SetEventMask procedure II-70
SetFilelnfo function
  high-level II-95
  low-level II-116
SetFillLock function
  high-level II-95
  low-level II-116
SetFileName function II-117
SetFileInfo function II-95
SetFlLock function II-95
SetFontLock procedure I-223
SetFPos function
  high-level II-93
  low-level II-111
SetGrowZone procedure II-42
SetHandleSize procedure II-34
SetItem procedure I-357
SetItemIcon procedure I-359
SetItemMark procedure I-359
SetItemStyle procedure I-360
SetText procedure I-422
SetTextIcon procedure I-359
SetTextMark procedure I-359
SetTextStyle procedure I-360
SetMaxCtl procedure I-327
SetMenuBar procedure I-355
SetMenuFlash procedure I-361
SetMFlash procedure I-361
SetMinCtl procedure I-326
SetOrigin procedure I-166
SetPBits procedure I-165
SetPenState procedure I-169
SetPort procedure I-165
SetPortBits procedure I-165
SetPt procedure I-193
SetPtSSize procedure II-37
SetRecRgn procedure I-183
SetRect procedure I-174
SetRectRgn procedure I-183
SetResAttrs procedure I-122
SetResFileAttrs procedure I-127
SetResInfo procedure I-122
SetResLoad procedure I-118
SetResPurge procedure I-126
SetSoundVol procedure II-233
SetStdProcs procedure I-198
SetString procedure I-468

SetTagBuffer function II-214
SetTime procedure II-380
SetTrapAddress procedure II-384
SetUpA5 procedure II-386
SetVol function
  high-level II-89
  low-level II-105
SetWindowPic procedure I-293
SetWordBreak procedure I-390
SetWRefCon procedure I-293
SetWTitle procedure I-284
SetZone procedure II-31
SEvtEnb global variable I-443
SFGetFile procedure I-523
SFPGetFile procedure I-526
SFPutFile procedure I-523
SFPutFile procedure I-519
SFRReply data type I-519
SFSaveDisk global variable I-519
SFTypeList data type I-523
ShieldCursor procedure I-474
ShowControl procedure I-322
ShowCursor procedure I-168
ShowHide procedure I-285
ShowPen procedure I-168
ShowWindow procedure I-285
signature III-9
SignedByte data type I-78

size
  of parameters I-90
  of variables I-85

size box I-287  See also grow region

size correction II-24

Size data type II-18
SizeControl procedure I-326
SizeResource function I-121
SizeRsrc function I-121
SizeWindow procedure I-290
SlopeFromAngle function I-475
socket II-265
socket client II-265
socket listener II-265
  writing your own II-324, 329
socket number II-265
socket table II-265

software overrun error II-246

sound buffer II-233; III-18, 21
Sound Driver I-13; II-221
  hardware II-233
  routines II-231
sound generator II-223; III-20
sound procedure I-409, 411, 425
Inside Macintosh

SoundBase global variable III-21
SoundDone function II-232
SoundLevel global variable II-234
SoundPtr global variable II-227
source transfer mode I-157
space state II-246
SpaceExtra procedure I-172
SPAlarm global variable See parameter RAM
SPATalkA global variable See parameter RAM
SPATalkB global variable See parameter RAM
SPClickCaret global variable See parameter RAM
SPConfig global variable II-305
speaker volume II-232, 371
SPFont global variable See parameter RAM
SPKbd global variable See parameter RAM
split bar I-49
SPMisc2 global variable See parameter RAM
spool printing II-151, 153
SPortSel data type II-249
SPPortA global variable See parameter RAM
SPPortB global variable See parameter RAM
SPPrint global variable See parameter RAM
SPValid global variable See parameter RAM
SPVolCtl global variable See parameter RAM
square-wave synthesizer II-223, 225
stack I-73; II-17
stack-based routines I-90
stack frame I-96; II-17
StageList data type I-424
stages of an alert I-409
Standard File Package I-12, 515
routines I-519
start bit II-245
StartSound procedure II-231
Stash function II-195
Status function
high-level II-179
low-level II-186
status information II-176
status routine of a driver II-187, 194
StdArc procedure I-199
StdBits procedure I-199
StdComment procedure I-199
StdGetPic procedure I-200
StdLine procedure I-198
StdOval procedure I-199
StdPoly procedure I-199
StdPutPic procedure I-200
StdRect procedure I-198
StdRgn procedure I-199
StdRRect procedure I-198
StdText procedure I-198
StdTxMeas function I-199
stillDown function I-259
stop bit II-245
StopAlert function I-419
StopSound procedure II-232
Str32 data type II-298
Str255 data type I-78
string comparison I-501, 506; II-376
string list I-468
resource format I-476
string manipulation I-468
StringHandle data type I-78
StringPtr data type I-78
StringToNum procedure I-490
StringWidth function I-173
structure region of a window I-271
StuffHex procedure I-195
style See character style
Style data type I-152
style dialog II-149
Style menu I-61
StyleItem data type I-152
SubPt procedure I-193
SWSynthPtr data type II-225
SWSynthRec data type II-225
synchronous execution
AppleTalk Manager II-273
Device Manager II-180
File Manager II-97
synthesizer buffer II-225
SysBeep procedure I-385
SysEdit function I-441
SysError procedure II-362
SysEvtMask global variable II-70
SysMap global variable I-114
SysMapHndl global variable I-114
SysParm global variable II-369
SysParmType data type II-370
SysPtr data type II-370
SysResName global variable I-114
system error alert II-357
system error alert table II-357, 359
System Error Handler II-13; II-18, 355
routine II-362
system error ID II-357
system event mask I-254; II-70
system font I-219
system font size I-219
system heap I-74; II-9
system resource I-103
system resource file I-103
system startup information II-120
system traps III-215
system window I-270, 438
SystemClick procedure I-441
SystemEdit function I-441
SystemEvent function I-442
SystemMenu procedure I-443
SystemTask procedure I-442, 444; II-189
SystemZone function II-32
SysZone global variable II-19, 21, 32

tag byte II-24
TEActivate procedure I-385
TECalText procedure I-390
TEClick procedure I-384
TECopy procedure I-386
TECut procedure I-385
TEDeactivate procedure I-385
TEDelete procedure I-387
TEDispose procedure I-383
TEDoText global variable I-391
TEFromScrap function I-389
TEGetScrapLen function I-389
TEGetText function I-384
TEHandle data type I-374
TEIdle procedure I-384
TEInit procedure I-383
TEInsert procedure I-387
TEKey procedure I-385
TENew function I-383
TEPaste procedure I-386
TEPtr data type I-374
TRec data type I-377
TRecal global variable I-391
TEScrapHandle function I-389
TESCroll procedure I-388
TEScrapHandle global variable I-389
TEScrapLength global variable I-389
TESetJust procedure I-387
TESetScrapLen procedure I-390
TSetSelect procedure I-385
TESetText procedure I-383
TestControl function I-325
TEToScrap function I-389
TEUpdate procedure I-387
text characteristics I-151
text in a dialog/alert I-404, 408
text streaming II-165
TextBox procedure I-388
TextEdit I-12, 371
routines I-383
scrap I-373, 388
TextFace procedure I-171
TextFont procedure I-171
TextMode procedure I-171
TextSize procedure I-171
TextWidth function I-173
TheMenu global variable I-357
thePort global variable I-162, 165
TheZone global variable II-31
thousands separator I-497
THPrint data type II-149
thumb I-312
THz data type II-22
tick I-246
TickCount function I-260
Ticks global variable I-260; II-198
Time global variable II-198, 369, 378
time operations II-377
ToExtFS global variable II-128
toggled command I-53, 357
Tone data type II-225
Tones data type II-225
Toolbox I-9
Toolbox Event Manager I-11, 241
routines I-257
Toolbox Utilities I-12, 465
routines I-467
ToolScratch global variable I-85
TopMapHndl global variable I-115
TopMem function II-44
TPPrint data type II-149
TPrPort data type II-147
TPrInfo data type II-150
TPrint data type II-149
TPrJob data type II-151
TPrPort data type II-147
TPrStatus data type II-161
TPrSt1 data type II-152
TPrXInfo data type II-152
track on a disk II-211
TrackControl function I-323
TrackGoAway function I-288
transaction II-266
transaction ID II-266
transaction release II-270
transaction request II-266
transaction response II-266
Transcendental Functions Package II-13; II-403, 407
transfer mode I-156
trap dispatch table I-87
routines II-383
trap dispatcher I-89
trap macro I-88, 90
list III-215
trap number I-89, 384
trap word I-88
TRel See transaction release
TReq See transaction request
TResp See transaction response
trigonometric functions II-407
type coercion I-79
type size See font size

UTableBase global variable II-192
Utilities, Operating System I-13; II-307
routines II-374
Utilities, Toolbox I-12, 465
routines I-467

V
validity status II-370
ValidRect procedure I-292
ValidRgn procedure I-292
variation code
control I-328
window I-298
VBL interrupt See vertical blanking interrupt
VBL task II-350
VBLQueue global variable II-352
VBLTask data type II-350
VCB data type I-125
VCBQHdr global variable II-126
vector II-196
vector table II-196
Versatile Interface Adapter III-39
version data III-10
version number of a file II-81
vertical blanking interrupt II-349; III-18
vertical blanking interval III-18
vertical retrace interrupt I-13; II-349
Vertical Retrace Manager I-13; II-347
routines II-351
vertical retrace queue II-350, 352
VHSSelect data type I-139
VIA III-39
VIA global variable I-198; III-39
VIA interrupts II-197; III-38, 41
video interface III-18
view rectangle I-374
VInstall function II-351
visible
control I-316
window I-274
visRgn of a grafPort I-149
volume (on a disk) II-79
volume (speaker) II-232, 371
volume allocation block map II-122
volume attributes II-121
volume buffer II-79
volume control block II-125
volume-control-block queue II-125
volume index II-102
volume information II-121
volume name II-79
volume reference number II-79
VRemove function II-351

W
WaitMouseUp function I-259
Wave data type II-227
waveform II-223
waveform description II-224
wavelength II-223
WavePtr data type II-227
white global variable I-162
window I-44, 269
closing I-45, 283
defining your own I-297
moving I-46, 289
opening I-45, 282
resource format I-302
sizing I-47, 289
splitting I-49
window class I-274, 276
window definition function I-272, 298
window definition ID I-273, 298
window frame I-271
window list I-274, 277
Window Manager I-11, 267
routines I-281
Window Manager port I-271, 282
window pointer I-275
window record I-274, 276
window template I-274
resource format I-302
WindowList global variable I-255, 277
WindowPeek data type I-275
WindowPtr data type I-275
WindowRecord data type I-276
WMgrPort global variable I-282
word I-42
in TextEdit I-373
word break routine I-380
word wraparound I-373
write data structure II-306
Write function, high-level
  Device Manager II-179
  File Manager II-92
Write function, low-level
  Device Manager II-185
  File Manager II-110
WriteDDP function II-312
WriteLAP function II-307
WriteParam function II-382
WriteResource procedure I-125

X
XorRgn procedure I-185

Y

Z
ZeroScrap function I-458
zone
  AppleTalk Manager II-266
  Memory Manager See heap zone
Zone data type II-22
zone header II-22
zone pointer II-22
zone record II-22
zone trailer II-22

III-281
Inside Macintosh

Welcome to the world of programming for the Macintosh®. No other personal computer has been as enthusiastically received by the programming community, as the large—and growing—body of Macintosh software attests. Inside Macintosh provides the guidelines and technical information that you’ll need to develop Macintosh programs, but many other resources can help speed and simplify your software development efforts.

Development Languages

You won’t have to look far to find a development language that suits your specific requirements. A growing family of Macintosh languages will serve your development needs whether your expertise is in Pascal, C, Assembler, FORTH, FORTRAN, COBOL, BASIC, Lisp, Modula-2, or one of many others. And the information in Inside Macintosh can be applied to any of the Macintosh languages.

The Certified Developer Program

If your primary business is developing software products for commercial markets, we strongly suggest that you investigate the Apple Certified Developer Program. This program helps developers produce and bring Macintosh products to market by providing them with support programs, services, and information. Among them are

- Technical Support: Apple’s Developer Technical Support Group offers fast answers by way of AppleLink® or MCI electronic mail.
- Macintosh Technical Notes: This is a bimonthly package of supplemental technical information.
- AppleLink: Through this electronic service, you can get answers to your technical questions and current information on Apple and third-party products and programs.
- Certified Developer Mailings: These monthly mailings keep you informed about Apple’s products, development tools, and technical and company directions.
- The Information Exchange: This information, available in printed and HyperCard® stack form, lists company-sponsored programs and services available to you and your company.
- Outside Apple: This monthly newsletter informs you of developer-oriented Apple groups, programs, and events.

You must meet certain criteria to get Certified Developer status. You can get an information package and application by writing to

Developer Programs
Apple Computer, Inc.
20525 Mariani Avenue, M/S 51-W
Cupertino, CA 95014

APDA

The Apple Programmer’s and Developer’s Association, APDA™, provides technical documentation and products for all programmers and developers who work on Apple equipment. It provides material that is unavailable elsewhere (including preliminary documentation of new Apple products). APDA also sells compilers and other tools from both Apple and third-party sources. For information on joining, write to

APDA
290 SW 43rd Street
Renton, WA 98055
(206) 251-6548

Technical Notes

Published bimonthly by Developer Technical Support, these notes answer frequently asked questions through examples and sample code and provide updates, additions, and corrections to the Inside Macintosh books. They are available through the Certified Developer Program, APDA, and major electronic information services.
Written by the people at Apple Computer, Inside Macintosh is the definitive source of information for programmers writing application programs, desk accessories, device drivers, and other software for any of the computers in the Apple Macintosh® family. It includes:

- Guidelines for designing a user interface that conforms to the Macintosh standard.
- Descriptions of more than 1,200 ROM- and disk-based routines.
- A description of the Macintosh hardware.

Inside Macintosh is your guide to creating software for the Macintosh, whatever programming language you use. It describes the Pascal interfaces to the routines and, wherever applicable, gives special information for programming in assembly language. (If you're using a high-level language other than Pascal, your development system documentation should tell you how to apply the information in Inside Macintosh.) A typical chapter describes a related set of routines, such as the Window Manager, and provides:

- Key concepts and background information.
- Hints on which routines you need to learn about and how they fit into your program.
- A detailed description of each routine.

Inside Macintosh consists of six volumes. This volume, Volume III, contains:

- A discussion of your program's interface with the Macintosh Finder.
- A description of the Macintosh 128K and 512K computers.
- Summaries of all the Managers and other software described in volumes I, II, and III.

Volume I contains important introductory material and describes the QuickDraw graphics package and important Managers such as the Resource, Font, and Menu Managers. Volume II complements Volume I in describing the Managers that perform such basic routines as file and device I/O, memory management, and interrupt handling. Volume IV discusses the changes introduced by the Macintosh 512K Enhanced and Macintosh Plus computers, including the Hierarchical File System and the SCSI port. Volume V discusses the changes introduced by the Macintosh SE and Macintosh II computers, including color, NuBus™ slots, and the Apple Desktop Bus. Inside Macintosh X-Ref provides a single index to Inside Macintosh and other Macintosh technical books.

About the cover: This design represents a new look for the original edition of Inside Macintosh, Volume III, and the other books in the Apple Technical Library. The contents have not been changed.