Programmer's Guide to Apple Scanners

Second edition

Developer Technical Publications

Apple Computer, Inc.
20525 Mariani Avenue
Cupertino, CA 95014
408-996-1010

Simultaneously published in the United States and Canada.

LIMITED WARRANTY ON MEDIA AND REPLACEMENT

If you discover physical defects in the manual or in the media on which a software product is distributed, APDA will replace the media or manual at no charge to you provided you return the item to be replaced with proof of purchase to APDA.

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

Figures and Tables ix

Preface About This Book xiii

What's in This Book xiii
How to Use This Book xiv
Conventions Used in This Book xiv
  Code Conventions xv
  Terminology xvi
Supplemental Reference Documents xvi
For More Information xvi

Chapter 1 A Scanning Primer 1

Anatomy of a Scan 2
  Pixels and Resolution 3
  About Scanning Parameters 4
    Composition 4
    Contrast 5
    Brightness 7
    Resolution 7
    Scan Area 8
    Gamma Correction 8
    Color Correction 8
    Automatic Background Adjustment 10
What a Scanning Application Needs to Do 10
  Initializing Scanning Parameters 11
  Defining an Image Buffer 11
  Initializing the Scan 11
  Retrieving the Image 11
  Storing the Image in an Output File 11
Apple Scanner Models 12

Chapter 2 The Apple Scanner Driver 13

About the Scanner Driver 14
Using the Scanner Driver 16
  Gaining Access to the Scanner 16
  Determining Scanner Capabilities 19
  Setting Scan Parameters 23

Starting the Scan and Reading Image Data 24
Using Advanced Scanner Features 27
Using the Scanner Driver From Assembly Language 29

Chapter 3 Scanner Driver Functions 31

Standard Functions 32
ScAbortScan 33
ScClose 34
ScDoScan 35
ScGetHalfTones 40
ScGetRes 41
ScGetStdFeatures 42
ScOpen 43
ScSetScanArea 44
Advanced Functions 45
ScGetAdvFeatures 46
ScGetButton 47
ScInvertPixels 48
ScLoadGamma 49
ScLoadMatrix 50
ScResetButton 51
ScSensorSelect 52
ScSetGraymap 53
ScSetGroup3 54
ScSetHTPattern 55
ScSetLamp 56
ScSetLED 57
ScSetNoCal 58
ScSetNoHome 59
ScSetScannerAtoD 60
ScSetSpeed 61
ScSetThreshold 62
ScSetWaitButton 63
ScVendorUnique 64

Chapter 4 Scanner Driver Data Structures 67

Standard Data Structures 68
ScAreaRec 69
ScCompRec 75
ScHalfToneArray 78
ScResArray 79
ScScanAreaRec 80

ScStdFeaturesRec 81
Advanced Data Structures 83
ScAdvFeaturesRec 84
ScPatRec 87
ScGammaTableRec 89
ScMatrix 90

Chapter 5 Scanner Driver Summary 91

Standard Constants 92
Advanced Constants 92
Standard Data Types 93
Advanced Data Types 95
Standard Functions 96
Advanced Functions 96
Function Result Codes 97
Device Manager Equivalents 100

Chapter 6 SCSI Interface for Apple Scanners 103

SCSI Hardware Interface 104
Connector assignments 104
Bus Phases 106
SCSI Implementation by Apple Scanners 107
SCSI Status Codes 108
SCSI Messages 108
Apple Scanner Message 108
OneScanner and Color OneScanner Messages 109

Chapter 7 SCSI Commands for Apple Scanners 111

TEST UNIT READY ($00) 113
REQUEST SENSE ($03) 114
REQUEST SENSE Return Structure Description for the Apple Scanner 115
REQUEST SENSE Return Structure Description for the OneScanner 117
REQUEST SENSE Return Structure Description for the Color OneScanner 119
INQUIRY ($12) 122
INQUIRY Return Structure Description for the Apple Scanner 123
INQUIRY Return Structure Description for the OneScanner 126
INQUIRY Return Structure Description for the Color OneScanner 129

MODE SELECT ($15) 133
MODE SELECT Parameter List Description 133
Apple-Specific Parameter Page Description for the Apple Scanner 134
Apple-Specific Parameter Page Description for the OneScanner 135
Apple-Specific Parameter Page Description for the Color OneScanner 137
Disconnect-Reconnect Parameter Page Description 138
RESERVE ($16) 140
RELEASE ($17) 142
MODE SENSE ($1A) 143
Apple-Specific Data Page Description for the Apple Scanner 144
Apple-Specific Data Page Description for the OneScanner 145
Apple-Specific Data Page Description for the Color OneScanner 147
SCAN ($1B) 149
SCAN Command Structure for the Apple Scanner 149
SCAN Command Structure for the OneScanner 150
SCAN Command Structure for the Color OneScanner 151
Window Parameter List 152
SEND DIAGNOSTIC ($1D) 153
DEFINE WINDOW PARAMETERS ($24) 154
DEFINE WINDOW PARAMETERS Command Structure for the Apple Scanner 154
DEFINE WINDOW PARAMETERS Command Structure for the OneScanner and the Color OneScanner 155
DEFINE WINDOW PARAMETERS Parameter List for the Apple Scanner 156
DEFINE WINDOW PARAMETERS Parameter List for the OneScanner 159
DEFINE WINDOW PARAMETERS Parameter List for the Color OneScanner 163
Using Window Descriptors With the Apple Scanner 166
GET WINDOW PARAMETERS ($25) 168
READ ($28) 170
READ Command Structure for the Apple Scanner 170
READ Command Structure for the OneScanner 171
READ Command Structure for the Color OneScanner 173
SEND ($2A) 176
SEND Command Structure for the Apple Scanner 176
SEND Command Structure for the OneScanner 177
SEND Command Structure for the Color OneScanner 178
Halftone Parameter Page Description for the Apple Scanner and the OneScanner 179
Gamma Data Write Format for the Color OneScanner 181
Color Correction Matrix for the Color OneScanner 182
PSRAM Data Write Format for the Color OneScanner 183
OBJECT POSITION ($31) 184

GET DATA STATUS ($34) 186
GET DATA STATUS Return Structure Description for the Apple Scanner 187
GET DATA STATUS Return Structure Description for the OneScanner and the Color OneScanner 189

Appendix A Specifications 193

Apple Scanner Specifications 193
Apple OneScanner Specifications 194
Apple Color OneScanner Specifications 196

Appendix B Optimizing the Color OneScanner 199

Optimizing the Scanner Matrix 199
Gamma Correction 200

Glossary 203

Index 207

Figures and Tables

Chapter 1 A Scanning Primer 1

Figure 1-1 A scanner block diagram 3
Figure 1-2 The effect of the contrast parameter 6
Figure 1-3 The effect of the brightness parameter in Line Art and Bi-level Color modes 7
Figure 1-4 The three options for the graymap parameter 9

Chapter 2 The Apple Scanner Driver 13

Table 2-1 Apple scanner driver standard functions 15
Table 2-2 Apple scanner driver advanced functions 15

Chapter 3 Scanner Driver Functions 31

Figure 3-1 Scanning into noncontiguous memory 36
Figure 3-2 How composition mode affects `rowBytes` and `byteWidth` 38

Chapter 4 Scanner Driver Data Structures 67

Figure 4-1 A scan rectangle 70
Figure 4-2 Scan areas at different resolutions 71
Figure 4-3 Orientation of returned data bits 74
Figure 4-4 Primary and secondary scan areas 85
Figure 4-5 Halftone processing 87
Figure 4-6 The electronic halftone matrix for a 4-by-4 spiral pattern 88

Table 4-1 Samples of halftone array elements 78

Chapter 5 Scanner Driver Summary 91

Table 5-1 Result codes 98
Table 5-2 Device Manager equivalents for standard driver functions 100
Table 5-3 Device Manager equivalents for advanced driver functions 101

Chapter 6 SCSI Interface for Apple Scanners 103

Table 6-1 SCSI connector pin assignments 104

Figure 7-1 The TEST UNIT READY command structure 113
Figure 7-2 The REQUEST SENSE command structure 114
Figure 7-3 The REQUEST SENSE return structure for the Apple Scanner 115
Figure 7-4 The REQUEST SENSE return structure for the OneScanner 117
Figure 7-5 The REQUEST SENSE return structure for the Color OneScanner 119
Figure 7-6 The INQUIRY command structure 122
Figure 7-7 The INQUIRY return structure for the Apple Scanner 123
Figure 7-8 The INQUIRY return structure for the OneScanner 127
Figure 7-9 The INQUIRY return structure for the Color OneScanner 130
Figure 7-10 The MODE SELECT command structure 133
Figure 7-11 The MODE SELECT Apple-specific parameter page for the Apple Scanner 134
Figure 7-12 The MODE SELECT Apple-specific parameter page for the OneScanner 135
Figure 7-13 The MODE SELECT Apple-specific parameter page for the Color OneScanner 137
Figure 7-14 The MODE SELECT disconnect-reconnect parameter page 139
Figure 7-15 The RESERVE command structure 140
Figure 7-16 The RELEASE command structure 142
Figure 7-17 The MODE SENSE command structure 143
Figure 7-18 The MODE SENSE Apple-specific data page for the Apple Scanner 144
Figure 7-19 The MODE SENSE Apple-specific data page for the OneScanner 145
Figure 7-20 The MODE SENSE Apple-specific data page for the Color OneScanner 147
Figure 7-21 The SCAN command structure for the Apple Scanner 149
Figure 7-22 The SCAN command structure for the OneScanner 150
Figure 7-23 The SCAN command structure for the Color OneScanner 151
Figure 7-24 The SCAN command Window identifier byte 152
Figure 7-25 The SEND DIAGNOSTIC command structure 153
Figure 7-26 The DEFINE WINDOW PARAMETERS command structure for the Apple Scanner 154
Figure 7-27 The DEFINE WINDOW PARAMETERS command structure for the OneScanner and the Color OneScanner 155
Figure 7-28 The DEFINE WINDOW PARAMETERS parameter list for the Apple Scanner 157
Figure 7-29 The DEFINE WINDOW PARAMETERS parameter list for the OneScanner 160
Figure 7-30 The DEFINE WINDOW PARAMETERS parameter list for the Color OneScanner 164
Figure 7-31 The GET WINDOW PARAMETERS command structure 168
Figure 7-32 The GET WINDOW PARAMETERS parameter list 169
Figure 7-33 The READ command structure for the Apple Scanner 170
Figure 7-34 The READ command structure for the OneScanner 171
Figure 7-35 The READ command structure for the Color OneScanner 173
Figure 7-36 Extra bit or byte returned for different resolutions and composition modes 175

Figure 7-37 The SEND command structure for the Apple Scanner 176
Figure 7-38 The SEND command structure for the OneScanner 177
Figure 7-39 The SEND command structure for the Color OneScanner 178
Figure 7-40 The SEND command halftone parameter page 179
Figure 7-41 The halftone matrix pattern for a 4-by-4 matrix 180
Figure 7-42 Gamma data write format 181
Figure 7-43 Color correction matrix 182
Figure 7-44 PSRAM data write format 183
Figure 7-45 The OBJECT POSITION command structure 184
Figure 7-46 The GET DATA STATUS command structure 186
Figure 7-47 The GET DATA STATUS return structure for the Apple Scanner 187
Figure 7-48 The GET DATA STATUS return structure 189

Table 7-1 SCSI commands and operation codes 112
Table 7-2 Transfer data types and transfer identification 174

Appendix A Specifications 193

Table A-1 The Apple Scanner specifications 193
Table A-2 The Apple OneScanner specifications 194
Table A-3 The Apple Color OneScanner specifications 196

Appendix B Optimizing the Color OneScanner 199

Table B-1 Color overlap matrix 200
Table B-2 Compensating for filter effects 200
Table B-3 Gamma values for Rasterops and Apple monitors 201

About This Book

The *Programmer's Guide to Apple Scanners* is for application developers who want to create an application capable of importing graphics data from the Apple Scanner, Apple OneScanner, or Apple Color OneScanner.

IMPORTANT

The Apple Scanner is always referred to in this book as the Apple Scanner. The Apple OneScanner is referred to as the Apple OneScanner, or more frequently as the OneScanner. The Apple Color One Scanner is referred to as the Apple Color OneScanner, or the Color OneScanner. The term "Apple scanners" refers generically to any one of the scanners covered by this publication. ▲

This book describes the program interface presented by the Apple scanner driver and the SCSI commands supported by Apple scanners. To use this book successfully, you must have some knowledge of a high-level programming language such as Pascal or C and experience with program development on the Apple Macintosh computer.

This book is not a user's guide or an owner's guide; for information on installing and operating your Apple Scanner, read the *Apple Scanner User's Guide.* For similar information on the Apple Color OneScanner and the Apple OneScanner, read the *Apple Color OneScanner and Apple OneScanner User's Guide.*

This preface describes the contents of each chapter and each appendix, and the typographical conventions used throughout the book. It also lists additional reference materials that will aid you in developing scanner application programs.

What's in This Book

This book consists of seven chapters, two appendixes, a glossary, and an index. The following list briefly describes the contents of each chapter and the appendixes. You can also use the table of contents and the index to help you locate specific topics.

- Chapter 1, "A Scanning Primer," provides a brief introduction to basic scanning concepts and terms. It summarizes the main features of the Apple Scanner, OneScanner, and Color OneScanner.
- Chapter 2, "The Apple Scanner Driver," provides background information about the Apple scanner driver, including a section on how to use it.

- Chapter 3, “Scanner Driver Functions,” describes the driver’s standard and advanced functions.
- Chapter 4, “Scanner Driver Data Structures,” describes the driver’s standard and advanced data structures.
- Chapter 5, “Scanner Driver Summary,” summarizes the constant values, data types, and functions for the standard and advanced features available in the scanner driver.
- Chapter 6, “SCSI Interface for Apple Scanners,” describes the SCSI interface presented by the SCSI hardware on the Apple scanners, including the supported bus phases.
- Chapter 7, “SCSI Commands for Apple Scanners,” describes the SCSI commands recognized by the Apple scanners.
- Appendix A, “Specifications,” provides tables containing specifications for the Apple Scanner, the OneScanner, and the Color OneScanner.
- Appendix B, “Optimizing the Color OneScanner,” provides information for improving the output images of the Color OneScanner.

How to Use This Book

Depending upon your requirements, you may not need all the information in this book. Follow these guidelines to make the most of this book:

- If you are not planning to create application programs and are reading this book to learn more about the way that scanners (particularly the Apple Scanner, OneScanner, and Color OneScanner) work, read Chapter 1.
- If you plan to write application programs for Apple scanners on the Macintosh computer, read Chapters 2 through 5 to acquaint yourself with the driver interface.
- If you plan to write application programs for Apple scanners on other computers, read Chapters 6 and 7 to learn about the SCSI commands supported by Apple scanners.

Conventions Used in This Book

The following standard warnings, cautions, and other visual cues are used throughout this book.

Note
A note like this contains supplementary information. ◆

IMPORTANT
This type of note contains information that is essential for an understanding of the main text. ▲

▲ **WARNING**
A warning like this directs your attention to something that could damage software or hardware, or that could result in loss of data. ▲

Terms in ***boldface*** type are defined in the glossary.

A special font, `Courier`, is used for characters that you type, or for lines of program code:

```
It looks like this.
```

A dollar sign precedes hexadecimal numbers, except in tables that clearly label hexadecimal numbers. For example, the hexadecimal equivalent of decimal 16 is written as $10.

The names of SCSI commands are in uppercase letters. For example, the SCAN command initiates a scan operation.

Although you can create an application program by using any one of several available languages, application code segments and driver code segments in this book are written in Pascal. Unless otherwise specified, all code references are to Pascal.

Code Conventions

Function declarations in this book follow a standard format. A typical function looks like this:

```
FUNCTION Filter (message: Integer; theBitmap: Bitmap;
      bounds: Rect; VAR flag: Boolean; selfHandle:
      Handle) : LongInt;
```

The names of functions, procedures, parameters, structures, and data types are in the `Courier` font. Words consisting of all uppercase letters are defined by the Pascal language (for example, `FUNCTION`). Words that contain both uppercase and lowercase letters with an initial capital letter identify data types that are defined by Apple Computer, Inc., or by your application. Words that contain both uppercase and lowercase letters but start with a lowercase letter identify variables, fields, or constants that are defined by Apple Computer, Inc., or by your application.

Terminology

Many terms used in this book are specific to computer graphics, and some are specific to the Apple Macintosh environment. The glossary contains the definitions of many terms found in this book. For your convenience, here are definitions of some important basic terms:

- ***host computer:*** A computer with a SCSI port that is connected to an Apple scanner.
- ***scanner:*** Generally, any graphic input device that converts printed matter into digital data. In this book, unless otherwise specified, this term refers to the Apple scanners.
- ***application program:*** A program on the host computer that sends commands and parameter data to, and receives image data from, the scanner.
- ***scanner driver:*** Software on the host computer that provides applications with a simplified interface to the Apple scanners.

The Apple Scanner is referred to in this text as the Apple Scanner, the Apple OneScanner as the OneScanner, and the Apple Color OneScanner as the Color OneScanner.

Supplemental Reference Documents

These Apple publications can help you understand Macintosh computer graphics, including QuickDraw and Color QuickDraw pictures:

- *Inside Macintosh*, Volumes I and IV, contain detailed information about QuickDraw.
- *Inside Macintosh*, Volumes V and VI, contain detailed information about Color QuickDraw.

These publications, as well as many reference documents, are available from APDA.

For More Information

APDA (Apple Programmers and Developers Association) is Apple's worldwide source for over 300 development tools, technical resources, training products, and information for anyone interested in developing applications on Apple platforms. Customers receive the quarterly *APDA Tools Catalog*, featuring all current versions of Apple, and most popular third-party development tools. Ordering is easy, there are no membership fees, and

application forms are not required for most APDA products. APDA offers convenient payment and shipping options, including site licensing.

To order a product or to request a complimentary copy of the *APDA Tools Catalog*, contact:

APDA
Apple Computer, Inc.
P.O. Box 319
Buffalo, NY 14207-0319

800-282-2732 (United States)
800-637-0029 (Canada)
716-871-6555 (International)
716-871-6511 (Fax)
AppleLink address: APDA
America Online: APDA
CompuServe: 76666m2405
Internet: APDA@applelink.apple.com

If you provide commercial products and services, call 408-974-4897 for information on the developer support programs available from Apple Computer, Inc.

CHAPTER 1

A Scanning Primer

The process of digitizing an image for use with a computer is called ***scanning.*** The device that performs this digitization is called a ***scanner.*** Scanners play an important role in any desktop publishing strategy.

The world of scanning is available to anyone with a scanner, a document, and a need to input graphics data into a Macintosh computer. The Apple Scanner, OneScanner, and Color OneScanner, with their simple command interfaces, make scanning available to all developers who wish to provide the capability in their application programs.

This chapter introduces you to basic image-scanning concepts and scanner hardware. It provides a high-level description of the way that the Apple scanners and scanning application programs work together.

Anatomy of a Scan

The scanning process consists of several steps: sensing the ***image*** on paper, placing the representative ***pixels*** in computer memory, and displaying the image on the screen. When a document is placed on the ***scanner glass,*** the scanner digitizes the image, and the scanner application places the image in an image file that can be imported by a graphics application such as MacPaint®. You can then manipulate the image as desired.

A scanner can also translate characters on paper into a text file. ***Optical character recognition*** (OCR) software recognizes text characters in scanned input and converts them to machine-readable form. Users can then work with that text just as they would any other text file.

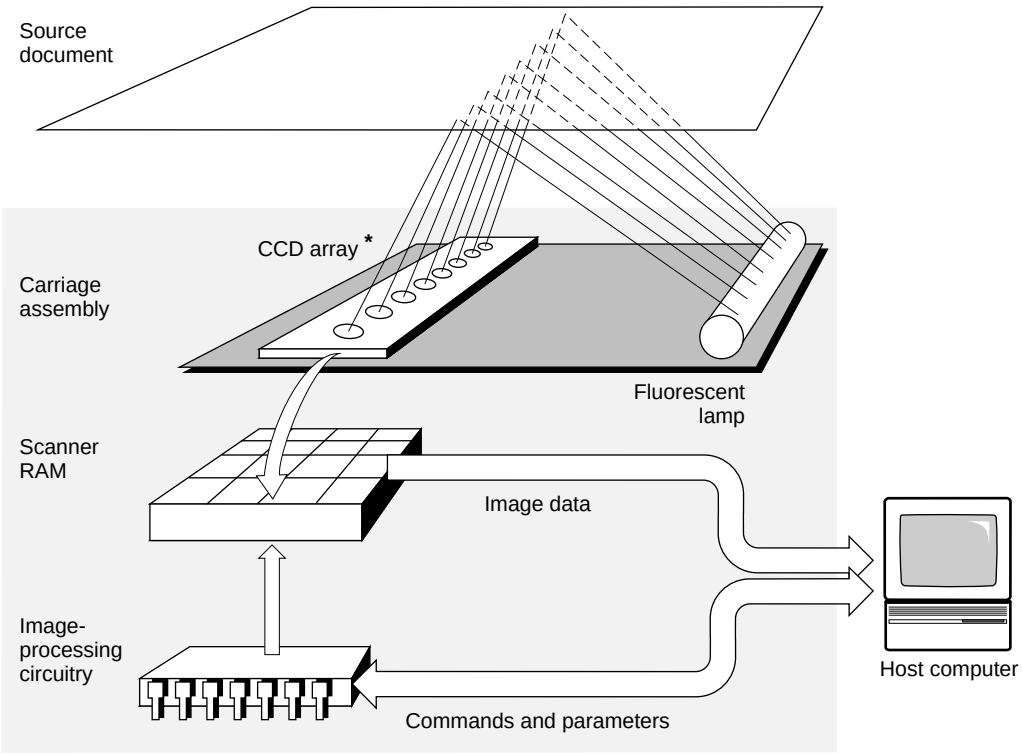
All Apple scanners are capable of digitizing any document, whether it is a graphics image or a page of text. Figure 1-1 is a simplified diagram showing the major components of the scanners that work together to accomplish this task.

Appendix B, "Optimizing the Color OneScanner," provides supplementary information on this subject.

A scanner is a self-contained unit that uses fluorescent light for the purpose of digitally encoding a source document. The scanner hardware senses the light reflected from the source document and stores it as digital data. The scanner then sends the scanned information, in digital format, to the host computer. Once the scanned document resides on the host computer, you can edit, print, display, or send the document through the AppleFax modem or through the AppleTalk network.

The application program running on the host computer sends commands and ***parameters*** to the scanner, setting scanning values within the scanner or instructing the scanner to perform some specific function. The scanner then performs a scan and sends the image data to the host computer, which stores it in a memory ***buffer*** until the data is viewed, printed, or saved to a disk file.

Figure 1-1 A scanner block diagram



* CCD = charge-coupled device (see Glossary)

Pixels and Resolution

To understand the scanning process, think of a source document as an array of dots or pixels (picture elements). This is exactly how the document appears to the scanner. The scanning ***resolution,*** controlled by the application program, determines the size of the dots. Resolutions as high as 300 dpi (dots per inch) are available. (For more information, see "Resolution," later in this chapter.) When scanning at 300dpi, a scanner interprets an 8.5-by-11-inch document as an array of more than 8,000,000 dots. This array is called a ***bitmap,*** and it is a digitized image of the ***original*** document. Depending upon other ***parameters*** that you specify, each dot in the bitmap is either black, white, a shade of gray, or a color.

Note

The terms "dots" and "pixels" are used interchangeably in this publication. ◆

The scanner creates the bitmap by passing a ***fluorescent lamp*** along the document and sensing the reflected light. The lamp is mounted on a movable ***carriage.*** The application program can designate any area of the document as the scanning area. It is within this area that the scanner digitizes images and creates a bitmap for use in the host computer.

About Scanning Parameters

Your application must provide the scanner with the parameters for scanning a document. These parameters determine the characteristics of the scan. The fundamental parameters required by a scanner are:

- Composition
- Contrast
- Brightness
- Resolution
- Scan area

Optional parameters are:

- Graymap (gamma correction)
- Color correction (Color OneScanner only)
- Automatic background adjustment

The host computer sends the parameters to the scanner via scanner commands. One or more parameters accompany each command sent to the scanner. The scanner uses each parameter to control some aspect of the scanning environment. This process provides a user with great flexibility. By modifying each of the parameters through the application, you can scan virtually any type of document and create an accurate representation of the original.

Composition

The ***composition parameter*** determines both the mode in which the scanner scans the image and the type of image data that the scanner returns. Each mode results in a different representation of the scanned image. The user chooses a mode after considering the nature of the original document and the planned use of the resulting image. The Apple scanners support a variety of modes. The Apple Scanner and the OneScanner support Line Art, Halftone, and Grayscale modes. The Color OneScanner supports Line Art, Grayscale, Bi-level Color, and Full Color modes.

When working in ***Line Art mode,*** the scanner places a black or white dot in the bitmap for each dot sensed on the original document. In this mode, the scanner also interprets shades of gray as either black or white. The value of the dot (black or white) is determined by the setting of the brightness parameter, which is described in "Brightness," later in this chapter.

Halftone mode generates an image by using halftone patterns to simulate shades of gray in the original document. A ***halftone pattern*** consists of a matrix of brightness values. In Halftone mode, the scanner filters scanned data through that matrix, turning result pixels on or off according to the brightness of the scanned image pixel relative to the

appropriate matrix value. The resulting two-level data emulates grayscale encoding in much the same way that dithering a color image fools the eye into seeing colors that are not there.

The user can use the halftone patterns that are built into the scanner or download one to the scanner. The scanner interprets each gray dot on the document as black or white using the halftone matrix as well as the contrast and brightness parameters, which are described next in "Contrast" and "Brightness."

Grayscale mode generates a gray dot in the bitmap for each dot on the original document. Different scanners have different gray-scale capabilities. The Apple Scanner supports 16 shades, or levels, of gray. The OneScanner supports 256 intensity levels. The content of the resulting image depends on the settings of the contrast and brightness parameters, which are described next in "Contrast" and "Brightness."

Bi-level Color mode produces 8-color bitmaps. Although scanning is in color, each color component, red, green and blue, has only two possible values: on or off.

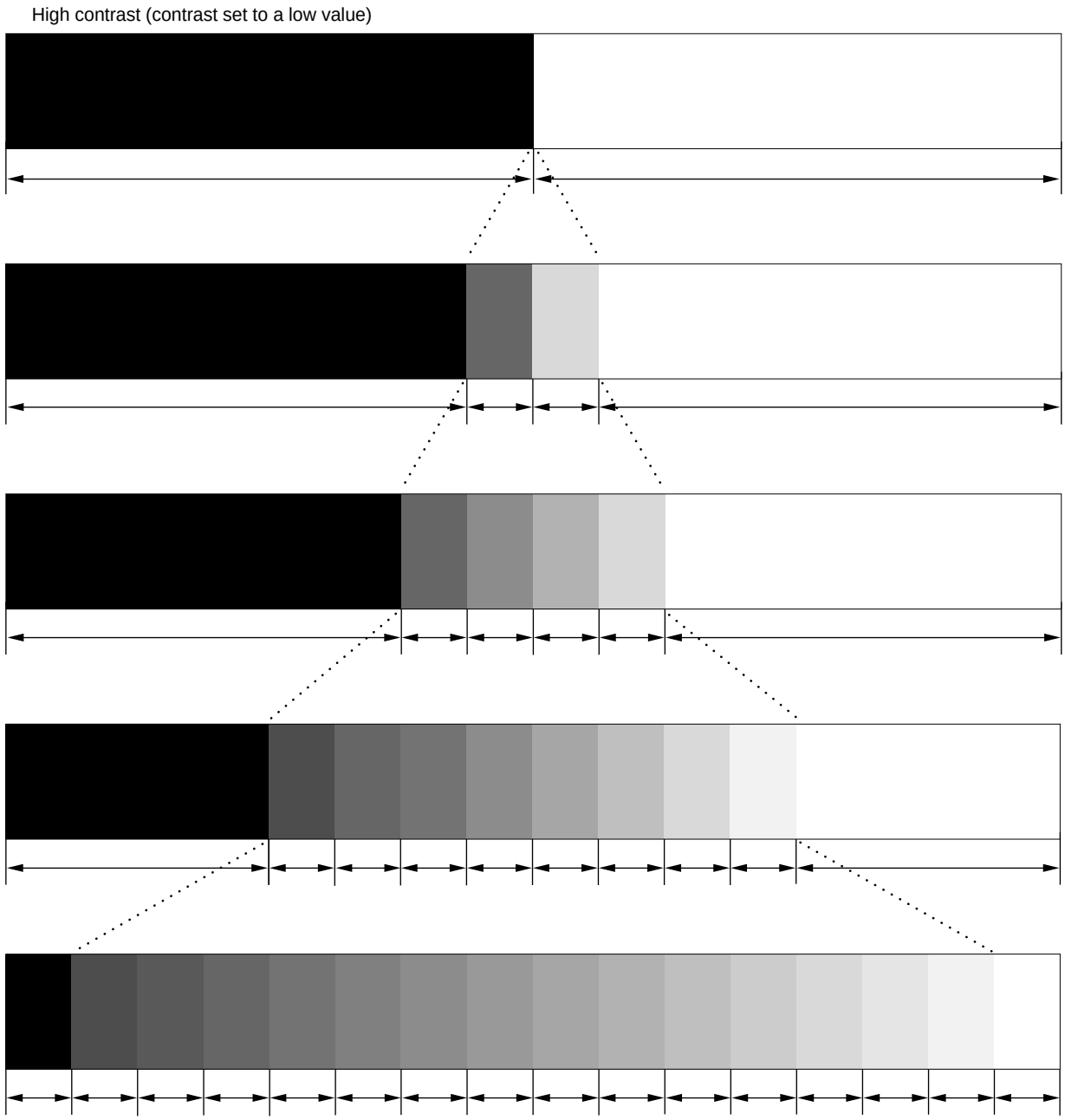
Full Color mode generates the largest range of colors. For each line, the scanner outputs color pixel data in three planes: red, green, and blue. Each of the three color components has 256 possible values (8 bits of each color), for a total of 16.8 million colors (3 colors x 8 bits = 24-bit color depth per pixel).

Contrast

The ***contrast parameter*** is used only in Grayscale and Full Color modes. It controls the intensity levels that the scanner applies to the scanned image. The value specified for the contrast parameter controls the amount of the visible spectrum that the scanner interprets as levels of gray. Those portions of the visible spectrum that fall outside the gray area are rendered as black or white, depending upon their placement in the spectrum. In Line Art and Bi-level Color modes, the scanners assume that this parameter has a value of 0 (off).

Low contrast values instruct the scanner to render only a small portion of the spectrum as intensity levels and a large portion as black or white, yielding an image with more pronounced (higher) contrast. Higher contrast values render a greater amount of the visible spectrum as shades of gray, so the image looks "flatter," with less of the visible spectrum rendered as black or white. Figure 1-2 on page 6 shows the effects of changing the contrast parameter. The contrast feature in the OneScanner and Apple Scanner were implemented in the opposite way to the contrast feature in the Color OneScanner. For example, increasing the value of contrast results in a lighter, or "flatter" image in the Color OneScanner, but in the OneScanner and the Apple Scanner, the image will be darker.

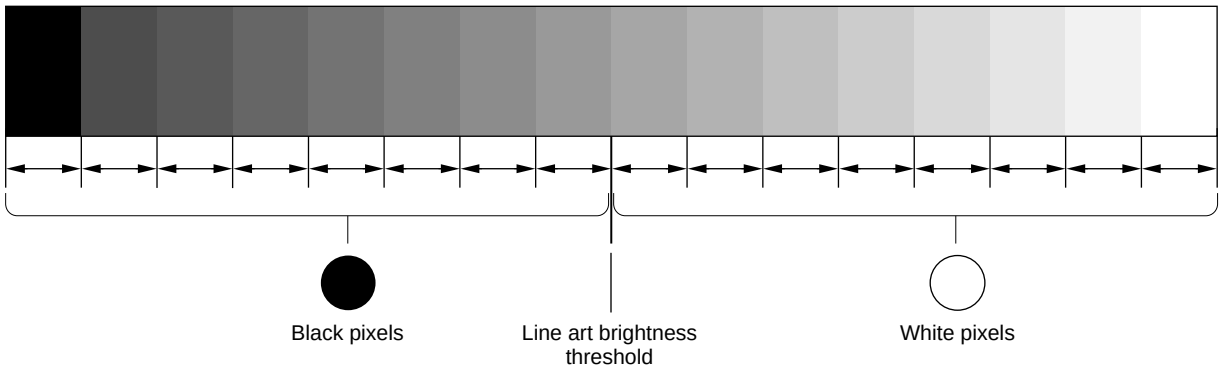
Figure 1-2 The effect of the contrast parameter



Brightness

The ***brightness parameter*** controls the brightness of the resulting image. The scanners interpret this value differently depending upon the composition mode. In Line Art mode, the brightness parameter determines the threshold level at which a dot goes from black to white. The scanner renders gray values that are greater than the brightness threshold as black pixels; it renders those that are less than the threshold as white pixels. Figure 1-3 shows how the brightness parameter determines which shades of gray result in black dots and which result in white dots.

Figure 1-3 The effect of the brightness parameter in Line Art and Bi-level Color modes



In Grayscale, Halftone, Bi-level Color, and Full Color modes, the brightness parameter positions the center point of the visible spectrum. The scanner then divides the two resulting parts into a number of intensity levels. Your application uses the contrast parameter to control the amount of the visible spectrum devoted to intensity levels. Higher brightness values yield a lighter image, because more of the visible spectrum is devoted to lighter shades of gray. Lower values yield a darker image.

In Bi-level Color mode, the brightness control shifts the black/white threshold level. The threshold ranges from 1 (minimum) to 255 (maximum), in steps of 1.

Resolution

The ***resolution parameter*** determines the number of horizontal and vertical dots that make up the bitmap. The value of the resolution parameter specifies the number of dots per inch in the resulting image. Higher values result in images with greater detail. Storing high-resolution images also requires more host computer memory than storing low-resolution images.

Although several different output resolutions are available, all documents are scanned at a horizontal resolution of 300 dpi. If you want a lower horizontal resolution, your application can alter the resolution setting of the scanner. The image-processing circuitry of the scanner later combines or averages pixels. The scanner generates the vertical

resolution by moving the scanning carriage at one of several speeds along the vertical axis of the document and by discarding lines of pixels to obtain a lower vertical resolution.

Alternatively, your application can leave the scanner at its highest resolution setting and later generate the lower-resolution data itself. Your application can reduce the resolution of the scanned image by averaging the values of adjacent pixels. By following this approach, you can determine the averaging algorithm used to generate the lower-resolution image. However, this technique is slower than relying on the scanner circuitry.

Scan Area

The ***scan area parameter*** defines the rectangular area on the source document for which the scanner generates a bitmap. With the Apple Scanner, you may define more than one scan area.

Gamma Correction

To the human eye, dark objects seem to lose detail. The Apple Scanner and the Color OneScanner provide correction techniques, called ***gamma correction,*** to compensate for this aspect of human vision. By applying the appropriate option, you can cause the scanner to accentuate the detail in either light or dark areas of the scanned image. Three correction options are available in the Apple Scanner to help minimize the effect of this optical illusion. Appendix B, "Optimizing the Color OneScanner," provides more information on this subject.

As Figure 1-4 shows, these three options for the ***graymap parameter*** allow Apple scanners to

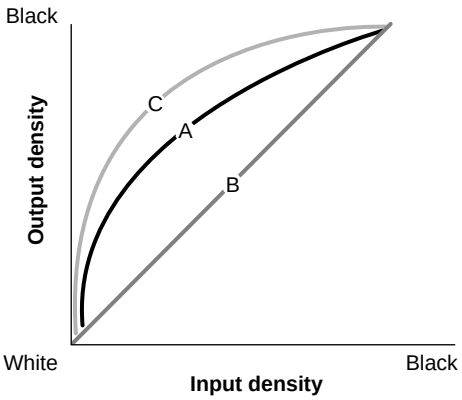
- simulate the visual response of the human eye (represented by line A)
- lighten the image, thereby enhancing the visible detail in the dark areas of the document and losing some detail in the light areas (represented by line B)
- darken the image, thereby enhancing the visible detail in the light areas of the document and losing some detail in the dark areas (represented by line C)

The OneScanner does not support gamma correction.

Color Correction

Color correction changes color data obtained from one source for output to a destination device. It is typically applied so that data looks good when displayed on a monitor or when printed. Different color correction values may be used according to the objective. For example, one objective might be to correct the data so that when it is displayed, it matches an original photograph. Another objective might be to correct the data so that when it is displayed, it matches the appearance of the printed data. The Color OneScanner incorporates a hardware mechanism, called the 3-by-3 matrix multiplier,

Figure 1-4 The three options for the graymap parameter



that can be loaded with the desired color correction. The color correction circuitry performs a 3-by-3 matrix calculation, as shown below:

$[R_{OUT}] = [A]\ [R_{IN}]$

$[G_{OUT}] = [A]\ [G_{IN}]$

$[B_{OUT}] = [A]\ [B_{IN}]$

where R, G, and B represent the colors red, green, and blue, IN is the input image, and OUT is the output image, and A represents the downloadable 3-by-3 matrix multiplier, with the following values:

00	01	02
10	11	12
20	21	22

The 3-by-3 matrix multiplier modifies the RGB values obtained by the scanner before the data is returned to the CPU. It performs a variety of functions:

- It adjusts the scanned image data for a particular output device, such as a printer, so that the image data, when printed, closely matches the original.
- It converts color data to gray-scale. This is done by loading the 3-by-3 matrix with the appropriate values. The de facto standard true luminance values for RGB are:

  Luminance = 0.30 ∗ R + 0.59∗ G + 0.11∗ B

  To set the values of the matrix, the application needs to determine the coefficients, using the following procedure. Since the multiplier is 16 bits, including the sign bit, and the range of the normalized values in the 3-by-3 matrix is 0 to 1, the coefficients in decimal are:

Red	.3 ∗ 32767	= 9830
Green	.59 ∗ 32767	= 19332
Blue	.11 ∗ 32767	= 3604

This gives a matrix of:

9830	19332	3604
9830	19332	3604
9830	19332	3604

Appendix B, "Optimizing the Color OneScanner," provides more information on this subject.

Automatic Background Adjustment

Your application lets the Apple Scanner automatically adjust the brightness threshold to bring out the most detail in a document. Called ***automatic background adjustment,*** this option can be used when one brightness setting does not provide the desired scanned result. By adjusting the brightness level automatically over the length of the document, automatic background adjustment can provide the best scanned image of a complex original document. The ***threshold parameter*** allows you to influence the operation of automatic background adjustment in the Apple Scanner.

The OneScanner and the Color OneScanner do not support automatic background adjustment.

What a Scanning Application Needs to Do

The scanner performs its functions unaided, but an application must first instruct it how and when to perform the scan. This section outlines the steps an application must take to initiate a successful scan of a document.

The scanner responds to commands and parameters sent to it by your application. The order in which the commands and data are delivered to the scanner is important; for example, your application must set the scanning parameters before sending a scan command. Your application must provide this information in the correct order and must be able to handle the image data returned from the scanner. To scan an original document successfully an application should:

1. Initialize the scanning parameters.
2. Define an image buffer within the host computer's memory.
3. Initiate the scan.
4. Retrieve the image data from the scanner and store it in the image buffer.
5. Save the data to a disk file (an optional step).

You can construct your application to perform any tasks required to fulfill the needs of the user, but it must perform the tasks described in the following sections.

Initializing Scanning Parameters

Initializing the scanning parameters establishes the context for the subsequent scan operation. All scanning parameters are initialized to a default value when you turn on the scanner, but you can customize the scanner for a particular image by specifying a value for each parameter. Your application may provide a mechanism for user control of the appropriate scanner parameters. Your application should retrieve the valid values for the scanner parameters from the scanner driver.

Defining an Image Buffer

The image buffer, located in the host computer's memory, stores the bitmap data that the scanner sends to the host computer. Once you specify the parameter values, your application can calculate the size of the bitmap required to store a scanned image. If the amount of memory available is too small to contain the complete bitmap, your application may write the surplus image data to a disk file. When the image is needed (for editing, display, or printing), the application can read the data from the disk file back into a buffer.

Note
The color format of the Color OneScanner is different from the QuickDraw format. You must therefore rearrange the data before QuickDraw can display it. See the section in Chapter 2, "Starting the Scan and Reading Image Data," for sample code that shows how data is rearranged. ◆

Initializing the Scan

When you prompt your application, it should send a command to the scanner to begin the scanning sequence. The scanner scans the document and stores the data in the scanner's internal memory.

Retrieving the Image

Once the scan begins, your application must poll the scanner to determine whether image data is available for transfer to the host computer. The scanner's memory is large enough to hold only a fraction of the image data. (See Appendix A, "Specifications.") Therefore, once your application starts receiving data, it must transfer image data repeatedly from the scanner's memory to the host computer's image buffer.

Storing the Image in an Output File

One reason to scan documents is to share image data. A standardized graphics file format simplifies data exchange. The Macintosh computer uses two common picture file formats. The Apple standard file contains a PICT data structure, which stores MacDraw pictures or bitmap data. The MacPaint picture file (PNTG) stores bitmap data. Your application may use either of these file formats to store the image data. It may also use any custom file format.

Apple Scanner Models

There are three members of the Apple scanner family, the Apple Scanner, the Apple OneScanner, and the Apple Color OneScanner. They are all flatbed scanners.

The Apple Scanner supports 4-bit gray-scale encoding (up to 16 levels of gray) and a number of resolutions ranging from 75 to 300 dots per inch (dpi). This is the original Apple scanner, and it is still supported by the current Apple scanner driver software.

The OneScanner supports 8-bit gray-scale encoding, providing up to 256 levels of gray. This scanner supports resolutions from 72 to 300 dpi, in increments of 1 dpi. The OneScanner also improves upon the Apple Scanner by providing faster data transfer speeds and a greater range of control options for scanning applications.

In Full Color mode, the Color OneScanner supports three (red, green, and blue) color planes, providing 256 possible values for each color component, with 16.8 million color choices. The Color OneScanner's Bi-level Color mode supports eight colors. The scanner supports resolutions from 72 to 300 dpi, in increments of 1 dpi.

Each scanner can accommodate source documents up to 8.5 inches wide by 14 inches long.

The Apple Scanner Driver

Graphics application programs communicate with a scanner in one of two ways: by sending commands and parameter data directly to the scanner, or by making calls to a scanner ***driver.*** A driver deals with much of the low-level work of handling commands and data, freeing the application to perform high-level tasks that are meaningful to the user. A driver may also isolate the application from interface differences between various supported scanners. The Apple scanner driver recognizes a predefined set of functions and, in turn, instructs the Apple scanners to perform the individual tasks that result in a scanned image.

This chapter discusses how your application can use the Apple scanner driver to control Apple scanners. If you are writing an application that runs on a Macintosh computer, use the functions for the Apple scanner driver listed in this chapter. If you are writing an application that runs on a computer other than a Macintosh computer, use the ***SCSI*** commands listed in Chapter 7, "SCSI Commands for Apple Scanners."

About the Scanner Driver

The Apple scanner driver provides a consistent, extensible interface to the Apple scanner family, including the Apple Scanner, the OneScanner, and the Color OneScanner. The driver isolates your application from many of the details of configuring a scanner and directing a scan operation, allowing you to focus on features that benefit the user. In addition, the driver interface provides an easy way for your application to determine the features of any attached Apple scanner and to take advantage of new scanner features in the future.

The driver provides a two-level interface, accommodating a wide range of graphics applications. The driver's standard functions support basic scanner features and allow an application to obtain a scanned image easily. These driver functions are most useful to applications that benefit from reading images directly from a scanner but do not need to control or manipulate the scanner itself. For example, draw and paint programs tend to concentrate on image creation and manipulation rather than scanner control. The standard driver functions are intended for such applications.

The advanced functions of the Apple scanner driver support applications that allow users to operate and control scanners. Such applications need to take advantage of all the features present in a particular scanner. The driver's advanced functions allow these programs to utilize the full range of features supported by a given scanner. The advanced functions include a routine that allows an application to determine the capabilities of the attached scanner.

To ensure consistency for future application development, the standard interface will change very little over time; any changes will always be compatible with older Apple scanner products. The advanced interface will grow constantly as Apple adds features to its scanner products. If you write applications that rely on these advanced features, your programs should always query the driver to determine which advanced features the attached scanner supports.

Table 2-1 summarizes the standard functions of the Apple scanner driver. Table 2-2 summarizes the advanced functions of the Apple scanner driver. Chapter 3, "Scanner Driver Functions," discusses each driver function in detail.

Table 2-1 Apple scanner driver standard functions

Standard function	Description
`ScAbortScan`	Cancels a scan in progress.
`ScClose`	Terminates the application's access to the scanner, freeing the device for use by another application.
`ScDoScan`	Initiates a scan and returns image data. Subsequent calls to `ScDoScan` retrieve further image data.
`ScGetHalfTones`	Retrieves information about the halftone patterns supported by the attached scanner.
`ScGetRes`	Retrieves information about the resolutions supported by the attached scanner.
`ScGetStdFeatures`	Retrieves information about the general capabilities of the attached scanner.
`ScOpen`	Grants application access to the attached scanner.
`ScSetScanArea`	Sets the scan area and parameters for a scan operation.

Table 2-2 Apple scanner driver advanced functions

Advanced function	Description
`ScGetAdvFeatures`	Retrieves information about the advanced capabilities of the attached scanner.
`ScGetButton`	Reads the state of the scanner button.
`ScInvertPixels`	Inverts pixel values. If `InvertFlag` is set to `TRUE`, it causes the value 255 to equal black, and 0 to equal full intensity.
`ScLoadGamma`	Loads tables of values to change the scanner's intensity curves.
`ScLoadMatrix`	Loads the matrix multiplier with the contents of the array data.
`ScResetButton`	Resets the state of the scanner button
`ScSensorSelect`	Allows the application to select which sensor on a color scanner is used for scanning gray.
`ScSetGrayMap`	Sets the graymap compensation curve for the attached scanner.
`ScSetGroup3`	Controls the image data compression technique used by the scanner.

continued

Table 2-2 Apple scanner driver advanced functions (continued)

Advanced function	Description
`ScSetHTPattern`	Downloads a custom halftone pattern to the attached scanner.
`ScSetLamp`	Controls the scanner lamp.
`ScSetLed`	Controls the scanner light-emitting diode (LED).
`ScSetNoCal`	Controls whether the scanner calibrates itself for the current lamp intensity before starting a scan operation.
`ScSetNoHome`	Controls whether the scanner returns the carriage assembly to the home position after a scan operation.
`ScSetScannerAtoD`	Allows the application to have direct access to the analog-to-digital converter used in the scanning process.
`ScSetSpeed`	Controls the data transfer speed between the scanner and the host computer.
`ScSetThreshold`	Sets the automatic background adjustment threshold level for the attached scanner.
`ScSetWaitButton`	Controls whether the scanner waits for the user to press the scanner button before starting a scan operation.
`ScVendorUnique`	Provides access to certain unique features supported by the attached scanner.

Using the Scanner Driver

This section discusses many of the basic steps involved in using the Apple scanner driver. Most of these topics deal with the standard driver functions and are therefore relevant to any scanning application. Some of these topics discuss more advanced subjects, including the advanced driver functions and the use of assembly language with the scanner driver. If you are not planning to develop an application that needs these features, you may want to skip the material relating to the advanced functions.

Gaining Access to the Scanner

By its nature a scanner can be used by only one application at a time. The scanner driver provides functions that allow your application to reserve and release the scanner. Before issuing any other driver functions, your application must gain control of the scanner by calling the `ScOpen` function. If another application has already reserved the scanner, this function returns a result code indicating that the scanner is in use. If the scanner is available, this function returns a reference number to your application. This reference number must be provided to all subsequent scanner driver functions.

When your application has finished using the scanner, it should release the device for use by other applications. Use the `ScClose` driver function to free the scanner.

The device driver is named .Scanner, and it is a standard Macintosh driver. If the scanner is switched on and the driver is properly installed, the system software automatically loads the scanner driver for you.

You should not open the scanner driver until you are ready to scan, and should close it immediately after every scan. Remember that you must reinitialize the scanning parameters each time you open the driver. Also be aware that another application may currently be using the scanner, in which case, you will receive the ScOpen error number "23."

The Apple scanners must be attached to the SCSI bus, switched on, and initialized before they can receive and process any commands from the host computer. ◆

The following sample code:

1. Opens the scanner driver.
2. Calls a routine to set the scanning parameters, which specify the area to be scanned and the characteristics of the area.
3. Calls another routine to scan an image.
4. Closes the scanner driver.

Note
The application that calls the `DoScanCommand` procedure must provide a formatted `ScStdFeaturesRec` record. The fields in this structure indicate the capabilities of the scanner driver, and the structure is set up by the `ScGetStdFeatures` routine, described later. ◆

```
PROCEDURE DoScanCommand(
   destPort:         GrafPtr;          { Put scanned image into this port }
   scannerInfo:      ScStdFeaturesRec; { Info about scanner features }
   compositionMode:  Integer;          { Desired composition mode }
   resolution:       Integer;          { Desired resolution }
   scanRectangle:    Rect;             { Scan this part of scanner bed }
   brightnessLevel:  Integer;          { Desired brightness level }
   contrastLevel:    Integer;          { Desired contrast level }
   pixelDepth:       Integer;          { Number of bits per pixel }
   ditherPattern:    Integer);         { Dithering pattern to use }

   CONST
      rCantOpenScanner  = 1;  { Resource ID of "Can't Open Scanner" string }
      rCantSetupScan    = 2;  { Resource ID of "Can't Setup Scan" string }
      rCantScanImage    = 3;  { Resource ID of "Can't Scan Image" string }
      rCantCloseScanner = 4;  { Resource ID of "Can't Close Scanner" string }
```

```
   VAR
      scanRef:    Integer;          { Scanner driver reference number }
      scanParms:  ScScanAreaRec;    { Parameters for upcoming scan }
      error:      OSErr;            { Code of any returned error }

   PROCEDURE HandleError(
         errorNum:      OSErr;
         messageNum:    LongInt);
      CONST
         rErrAlert         = 128;   { Resource ID of error alert }
         rErrStringList    = 129;   { Resource ID of error strings }
      VAR
         messageString:    Str255;  { String containing error mesage }
         result:           Integer; { Result of alert; ignored }

   BEGIN
      { Make sure scanner driver is closed }
      IF scanRef <> 0 THEN
         error := ScClose( scanRef );
      { Tell the user what happened }
      IF errorNum = openErr THEN
         BEGIN
            GetIndString( messageString, rErrStringList, messageNum );
            ParamText( messageString, '', '', '' );
            result := Alert( rErrAlert, NIL );
         END;
      EXIT( DoScanCommand );
   END;

BEGIN
   scanRef := 0;
   { Open scanner driver; get returned scanner reference number }
   error := ScOpen( scanRef );
   IF error <> noErr THEN
      HandleError( error, rCantOpenScanner );
   { Set the scanning parameters in preparation for the scan }
   error := SetUpScan(
         scanRef,
         scannerInfo,
         compositionMode,
         resolution,
         scanRectangle,
         brightnessLevel,
```

```
            contrastLevel,
            pixelDepth,
            ditherPattern,
            scanParms );
      IF error <> noErr THEN
         HandleError( error, rCantSetupScan );
      { Scan the image }
      error := ScanImage(
            scanRef,
            destPort,
            scannerInfo,
            scanParms.scanAreas[1] );
      IF error <> noErr THEN
         HandleError( error, rCantScanImage );
      { Image retrieved; now close the scanner driver }
      error := ScClose( scanRef );
      IF error <> noErr THEN
         HandleError( error, rCantCloseScanner);
END;
```

Determining Scanner Capabilities

The Apple scanner driver supports the entire family of Apple scanners. Each scanner provides different features and functions. Consequently, software that uses the Apple scanner driver should be self-configuring and should use the scanner driver's routines to determine the capabilities of the attached scanner. This applies to even the most basic scanning applications. If your program tries to use features that are not supported by the attached scanner, the scanner may not behave predictably.

The scanner driver provides a number of standard functions that supply information about the capabilities of the attached scanner. The `ScGetStdFeatures` function allows your application to determine the basic capabilities of the attached scanner, including the size of the scanning bed, supported composition modes, and valid values for certain scanning parameters such as brightness and contrast. The `ScGetRes` function returns information about the resolution values supported by the scanner. The `ScGetHalfTones` function returns a list of the supported halftone patterns. You may use the function `ScGetAdvFeatures` to find out which advanced features are supported. (See the section "Using Advanced Scanner Features," later in this chapter.)

Immediately after gaining access to the scanner (with the `ScOpen` function), your application should gather information about the attached scanner by calling all these routines. Your program can then use the returned information to build appropriate scrolling lists and menu items for the attached scanner. By querying the driver for this information, your application can automatically take advantage of the features and capabilities of new devices.

The following sample code retrieves information about the attached scanner by calling the `ScGetStdFeatures`, `ScGetRes`, and `ScGetHalfTones` driver functions.

```
FUNCTION GetScannerInfo(
      VAR scannerInfo:     ScStdFeaturesRec; { Returns scanner features }
      VAR lineArtRes:      ScResPtr;         { Returns line-art resolutions }
      VAR halfToneRes:     ScResPtr;         { Returns halftone resolutions }
      VAR grayScaleRes:    ScResPtr;         { Returns grayscale res's }
      VAR bilevelColorRes: ScResPtr;         { Returns bilevel res's }
      VAR fullColorRes:    ScResPtr;         { Returns full color res's }
      VAR halfToneNames:   ScHalfTonePtr)    { Returns halftone names }
   : OSErr;
VAR
      scanRef:          Integer;    { Reference number of scanner driver }
      compositionMode:  Integer;    { Composition mode for scanner features }
      resList:          ScResPtr;   { Handle to list of elements }
      numElements:      Integer;    { Number of elements in list }
      error:            OSErr;      { Error code of any error that occurs }

PROCEDURE HandleError(
      errorCode: OSErr );

   BEGIN
      { Close the driver if it has been left open }
      IF scanRef <> 0 THEN
         error := ScClose( scanRef );
      { Dispose of the resolution and halftone name arrays }
      IF lineArtRes <> NIL THEN
         DisposePtr( Ptr(lineArtRes) );
      IF halfToneRes <> NIL THEN
         DisposePtr( Ptr(halfToneRes) );
      IF grayScaleRes <> NIL THEN
         DisposePtr( Ptr(grayScaleRes) );
      IF bilevelColorRes <> NIL THEN
         DisposePtr( Ptr(bilevelColorRes) );
      IF fullColorRes <> NIL THEN
         DisposePtr( Ptr(fullColorRes) );
      IF halfToneNames <> NIL THEN
         DisposePtr( Ptr(halfToneNames) );
      { Make sure everything's returned NIL }
      lineArtRes := NIL;
      halfToneRes := NIL;
      grayScaleRes := NIL;
      bilevelColorRes := NIL;
```

```
      fullColorRes := NIL;
      halfToneNames := NIL;
      { Return the error code }
      GetScannerInfo := errorCode;
      EXIT (GetScannerInfo);
   END;

BEGIN
   { Prepare for any failure conditions }
   scanRef := 0;
   lineArtRes := NIL;
   halfToneRes := NIL;
   grayScaleRes := NIL;
   bilevelColorRes := NIL;
   fullColorRes := NIL;
   halfToneNames := NIL;
   { Open scanner driver; get returned scanner reference number }
   error := ScOpen( scanRef );
   IF error <> noErr THEN
      HandleError( error );
   { Ask scanner driver for scanner capabilities }
   error := ScGetStdFeatures( scanRef, @scannerInfo,
         SIZEOF (ScStdFeaturesRec) );
   IF error <> noErr THEN
      HandleError( error );
   { Get information for each composition mode }
   FOR compositionMode := scLineArt TO scFullColor DO
      IF scannerInfo.composition[compositionMode].resElements <> 0 THEN
         BEGIN
            { Allocate list of supported or preferred resolutions }
            numElements := scannerInfo.composition[compositionMode].
               resElements;
            resList := ScResPtr(NewPtr( numElements *
               SIZEOF (Integer) ));
            IF resList = NIL THEN
               HandleError( MemError );
            { Get the resolution list from scanner driver }
            error := ScGetRes( scanRef, compositionMode, resList );
            IF error <> noErr THEN
               HandleError( error );
            { Assign to the appropriate resolution list }
            CASE compositionMode OF
               scLineArt:        lineArtRes := resList;
```

```
            scHalfTone:       halfToneRes := resList;
            scGrayScale:      grayScaleRes := resList;
            scBiLevelColor:   bilevelColorRes := resList;
            scFullColor:      fullColorRes := resList;
         END;
      END;

   { Get the list of halftone names if halftone mode is supported }
   IF scannerInfo.composition[scHalfTone].resElements <> 0 THEN
      BEGIN
         { Allocate list of halftone patterns }
         numElements := scannerInfo.composition[scHalfTone].
               halfToneElements;
         halfToneNames := ScHalfTonePtr(NewPtr( numElements *
               SIZEOF (Str31) ));
         IF halfToneNames <> NIL THEN
            HandleError( MemError );
         { Get list of halftone names from driver }
         error := ScGetHalfTones( scanRef, scHalfTone, halfToneNames );
         IF error <> noErr THEN
            HandleError( error );
      END;
   { Close the scanner driver }
   error := ScClose( scanRef );
   { Indicate that this function finished normally }
   GetScannerInfo := noErr;
END;
```

The `ResolutionIsValid` function in the next code sample determines whether a particular resolution, specified by the `resolution` parameter, is available in the composition mode that the `scannerInfo` parameter represents. The `resList` parameter points to the array of resolutions for the composition mode. If the specified resolution is supported, then `ResolutionIsValid` returns `TRUE`; otherwise it returns `FALSE`. The `ResolutionIsValid` function supports both resolution ranges and lists of discrete resolutions.

```
FUNCTION ResolutionIsValid(
      resolution:    Integer;   { Resolution to check }
      scannerInfo:   ScCompRec; { Description of a composition mode }
      resList:       ScResPtr)  { Array of resolutions for composition mode }
   : Boolean;

   VAR
      validResolution:  Boolean;    { True if resolution found to be valid }
      index:            Integer;    { Index into resList }
```

```
BEGIN
   validResolution := FALSE;
   { Check to see if the desired resolution is OK }
   IF BTST (scannerInfo.resFlags, 0) THEN
      BEGIN
         { resList specifies a range of allowable resolutions }
         IF (resolution >= resList^[1]) AND
               (resolution <= resList^[scannerInfo.resElements]) THEN
            validResolution := TRUE;
      END
   ELSE
      BEGIN
         { resList is an array of allowable resolutions }
         index := 1;
         WHILE (NOT validResolution) AND
               (index <= scannerInfo.resElements) DO
            BEGIN
               validResolution := resolution = resList^[index];
               index := index + 1;
            END;
      END;
   ResolutionIsValid := validResolution;
END;
```

Setting Scan Parameters

After you have determined the scanner's features, set the scan area parameters using the `ScSetScanArea` function. The scan area parameters tell the scanner where and how a scan should take place. Your program must establish these parameter settings before it can start a scan operation. The values for all these parameters must meet the restrictions established for the attached scanner.

The following sample code sends the desired scanning parameters to the scanner. The `SetUpScan` function sets the composition mode, resolution, scanning rectangle, brightness, contrast, pixel depth, and halftone pattern based on your selection.

```
FUNCTION SetUpScan(
      scanRef:          Integer;
      scannerInfo:      ScStdfeaturesRec;
      renderMode:       Integer;
      resolution:       Integer;
      scanRectangle:    Rect;
      brightnessLevel:  Integer;
      contrastLevel:    Integer;
      pixelDepth:       Integer;
```

```
      ditherPattern:    Integer;
      VAR scanInfo:     ScScanAreaRec )
   : OSErr;

BEGIN
   { If brightness specified using new method, set high bit }
   IF scannerInfo.composition[renderMode].brightnessRange <> 0 THEN
      brightnessLevel := BOR (brightnessLevel, $8000);
   {If contrast specified using new method, set high bit }
   IF scannerInfo.composition[renderMode].contrastRange <> 0 THEN
      contrastLevel := BOR (contrastLevel, $8000);
   { Set up the ScScanAreaRec }
   scanInfo.reserved := 0;
   scanInfo.numAreas := 1;
   WITH scanInfo.scanAreas[1] DO
      BEGIN
         reserved := 0;
         xDpi := resolution;
         yDpi := resolution;
         scanRect := scanRectangle;
         brightness := brightnessLevel;
         contrast := contrastLevel;
         composition := renderMode;
         bitsPerPixel := pixelDepth;
         halfTone := ditherPattern;
      END;
   { Set the information for the upcoming scan }
   SetUpScan := ScSetScanArea( scanRef, @scanInfo );
END;
```

Starting the Scan and Reading Image Data

Once your application has properly configured the scanner for the scan operation, it starts the scan by calling the `ScDoScan` driver function. Subsequent calls to this function return image data from the scan. Your program should continue to call `ScDoScan` repeatedly until the scan is complete and all of the image data is transferred. The following sample code illustrates this process. The application:

1. Calculates the number of bytes that the scanner will return for every row of pixels that it returns.
2. Calculates the number of bytes that should be accepted from the scanner for every call to `ScDoScan`. This is always a multiple of `byteWidth`. If 24-bit color scanning is requested, a buffer is allocated to convert the 24-bit scanned image data to the 32-bit pixels that Color QuickDraw handles.

3. Enters the scanning loop, which places the scanned image data in the destination bitmap or pixel map. If a 24-bit color scan is being done, the scanned image data is put into the temporary buffer.
4. Places the image data from the temporary buffer in the destination pixel map, in the order compatible with Color QuickDraw.

When `ScanImage` is completed, `destPort` contains the complete, scanned image. A precondition of `ScanImage` is that the `scBilevelColor` mode not be used.

```
FUNCTION ScanImage(
      scanRef:       Integer;          { Driver reference number of scanner }
      destPort:      GrafPtr;          { GrafPort to scan into }
      scannerInfo:   ScStdFeaturesRec; { Features of the scanner }
      scanParms:     ScAreaRec)        { Specifications of scan }
   : OSErr;

   VAR
      destImage:        Ptr;        { Pointer to dest bit/pixel image }
      buffer:           Ptr;        { Buffer to hold planar color data }
      rovingBuffer:     Ptr;        { Pointer to each line in buffer }
      destBuffer:       Ptr;        { Pointer to each byte in destImage }
      destRowBytes:     Integer;    { # bytes in a row of bit/pixel image }
      byteWidth:        Integer;    { # bytes in a row to scan }
      dataLen:          LongInt;    { Length of data to read in bytes }
      minDataLen:       LongInt;    { Minimum dataLen for smooth scanning }
      numScannedRows:   Integer;    { Number of rows that were scanned }
      channel:          Integer;    { Index into the color channels }
      index:            Integer;    { Index into each color channel }
      rowNum:           Integer;    { Index into rows of the buffer }
      error:            OSErr;      { Return value for error condition }

BEGIN
   buffer := NIL;
   { Get pointer to destination bit/pixel image and its rowBytes }
   IF BAND (destPort^.portBits.rowBytes, $8000) <> 0 THEN
      BEGIN
         destImage := GetPixBaseAddr( CGrafPtr(destPort)^.portPixMap );
         destRowBytes := BAND (CGrafPtr(destPort)^.portPixMap^^.rowBytes,
               $3FFF );
      END
   ELSE
      BEGIN
         destImage := destPort^.portBits.baseAddr;
         destRowBytes := destPort^.portBits.rowBytes;
      END;
```

```
{ Calculate the number of bytes across the scan }
WITH scanParms DO
   IF scanParms.composition = scFullColor THEN
      byteWidth := ((((8 * (scanRect.right - 1)) DIV 8) + 1) -
            ((8 * scanRect.left) DIV 8)) * 3
   ELSE
      byteWidth := (((bitsPerPixel * (scanRect.right - 1)) DIV 8) + 1) -
            ((bitsPerPixel * scanRect.left) DIV 8);
{ Calc minimum dataLen; multiple of byteWidth and >= minReadSize }
WITH scannerInfo.composition[scanParms.composition] DO
   IF minReadSize > byteWidth THEN
      minDataLen := byteWidth * ((minReadSize - 1) DIV byteWidth + 1)
   ELSE
      minDataLen := byteWidth;
{ If scanning 24-bit color, allocate a buffer to rearrange pixel data }
IF scanParms.composition = scFullColor THEN
   buffer := NewPtrClear( minDataLen + byteWidth * 2 );
{ Loop until the scan is done }
error := noErr;
WHILE error = noErr DO
   BEGIN
      { Set maximum amount of data to read at minDataLen }
      dataLen := minDataLen;
      IF (error = noErr) AND (dataLen > 0) THEN
         BEGIN
            IF scanParms.composition = scFullColor THEN
               BEGIN
                  { Scan a piece of the image }
                  error := ScDoScan( scanRef, buffer, dataLen, 0,
                        0, 0 );
                  numScannedRows := dataLen DIV byteWidth;
                  rowNum := 1;
                  rovingBuffer := buffer;
                  WHILE (rowNum <= numScannedRows) DO
                     BEGIN
                        FOR channel := 1 TO 3 DO
                           BEGIN
                              FOR index := 0 TO (byteWidth DIV 3) - 1 DO
                                 BEGIN
                                    { Convert image to CQD format }
                                    destBuffer := Ptr(LongInt(destImage)
                                          + channel + index * 4);
                                    destBuffer^ := Ptr(LongInt(
```

```
                                        rovingBuffer) + index +
                                        (byteWidth DIV 3) * (channel -
                                        1))^;
                                 END;
                              END;
                           destImage := Ptr(LongInt(destImage) +
                                 destRowBytes);
                           rovingBuffer := Ptr(LongInt(rovingBuffer) +
                                 byteWidth);
                           rowNum := SUCC (rowNum);
                        END;
                     END
                  ELSE
                     BEGIN
                        { Scan a piece of the image }
                        error := ScDoScan( scanRef, destImage, dataLen, 0,
                              byteWidth, destRowBytes );
                        { If continuing scan, reposition destImage for more }
                        destImage := Ptr(LongInt(destImage) + (dataLen DIV
                              byteWidth) * destRowBytes);
                     END;
               END;
         END;
   { If error was scEOS (end of scan), then everything is OK }
   IF error = scEOS THEN
      error := noErr;
   { Get rid of the buffer used for 24-bit color }
   IF buffer <> NIL THEN
      DisposePtr( buffer );
   ScanImage := error;
END;
```

Using Advanced Scanner Features

Applications can also utilize the advanced features present on the attached scanner. As with standard features, your program should use the facilities provided by the scanner driver to determine which advanced features the scanner supports. The scanner driver's `ScGetAdvFeatures` function returns information about the advanced capabilities of the scanner, including information about which advanced driver functions are supported. After retrieving this information, your program can then proceed to use the advanced features of the scanner.

The following sample code illustrates the use of the `ScGetAdvFeatures` driver function. This routine, called `FlashLamp`, causes the scanner lamp to flash twice, for a period of four seconds each time. The application first determines whether

the ScSetLamp function exists by checking on the return result of ScGetAdvFeatures. It then calls ScSetLamp only if it is available on the connected scanner.

```
FUNCTION FlashLamp(
      scanRef:    Integer)    { Reference number of scanner driver }
   : OSErr;

   CONST
      kSetLampPresent = 3; { Bit indicating that ScSetLamp is available }

   VAR
      lastTicks:  LongInt;          { Tick count at end of Delay }
      specials:   ScAdvFeaturesRec; { Info about advanced scanner features }
      error:      OSErr;            { Error code from scanner driver }

   PROCEDURE HandleError(
         error: OSErr);

   BEGIN
      FlashLamp := error;
      EXIT (FlashLamp);
   END;

BEGIN
   { Get the advanced features of the scanner }
   error := ScGetAdvFeatures( scanRef, @specials,
         SIZEOF (ScAdvFeaturesRec) );
   IF error <> noErr THEN
      HandleError( error );
   { Check to see whether ScSetLamp is available }
   IF BTST (specials.controlFlags, kSetLampPresent) THEN
      BEGIN
         { Turn the lamp on }
         error := ScSetLamp( scanRef, true );
         IF error <> noErr THEN
            HandleError( error );
         { Wait four seconds }
         Delay( 240, lastTicks );
         { Now turn the lamp off }
         error := ScSetLamp( scanRef, false );
         IF error <> noErr THEN
            HandleError( error );
      END;
   FlashLamp := noErr;
END;
```

Using the Scanner Driver From Assembly Language

The device driver is named .Scanner, and it is a standard Macintosh driver. The scanner driver supports the `Open Driver`, `Close Driver`, `Control`, `Status`, and `Read` routines of the Macintosh Device Manager. Chapter 3 discusses each driver function, and provides assembly-language interface information.

If your application uses the assembly-language interface to the scanner driver, the interface code automatically loads the driver for you.

Scanner Driver Functions

This chapter describes each of the driver's standard and advanced functions that are available to applications. The data structures used are described in Chapter 4, "Scanner Driver Data Structures," and Chapter 5, "Scanner Driver Summary." If you are writing an application that runs on a Macintosh computer, use the functions for the Apple scanner driver listed in this chapter. If you are writing a scanner application that runs on a computer other than the Macintosh computer, use the ***SCSI*** commands listed in Chapter 7, "SCSI Commands for Apple Scanners."

Standard Functions

The standard scanner driver functions available to applications are sufficient to support most scanning applications.

Note
Many parameters sent to the driver or returned from the driver are listed as reserved. These parameters must be set to 0 if they are supplied to the driver. If they are parameters returned from the driver, they will always contain a value of 0. ◆

Standard functions that apply to all Scanners are:

- `ScAbortScan`
- `ScClose`
- `ScDoScan`
- `ScGetRes`
- `ScGetStdFeatures`
- `ScOpen`
- `ScSetScanArea`

The standard function that applies only to the Apple Scanner and the OneScanner is:

- `ScGetHalfTones`

ScAbortScan

The `ScAbortScan` function cancels the scan in progress, causing the scanner driver to discard any image data that the application has not yet read. While a scan is in progress, an application may call only the `ScAbortScan` and `ScDoScan` functions. Calling `ScAbortScan` when no scan is in progress has no effect.

PASCAL

```
FUNCTION ScAbortScan (refNum: Integer) : OSErr;
```

Assembly-Language Note

Calling `ScAbortScan` is equivalent to calling the Device Manager `Control` routine with `csCode` set to 1, which is a `killIO` control call. ◆

`refNum` Scanner identifier returned by the `ScOpen` function.

RESULT CODES

`noErr`	0	No error
`scComErr`	-17065	The communication interface is not operating properly
`scResetErr`	-17066	Scanner has been reset or reinitialized
`scScannerErr`	-17068	Internal scanner malfunction

ScClose

The `ScClose` function closes the scanner driver, making the driver available to other applications. Before quitting, your application should call `ScClose`, otherwise, the scanner remains unavailable for use by other applications.

PASCAL

```
FUNCTION ScClose (refNum: Integer) : OSErr;
```

Assembly-Language Note

Calling `ScClose` is equivalent to calling the Device Manager `Close` routine. ◆

`refNum` Scanner identifier returned by the `ScOpen` function.

RESULT CODES

`noErr`	0	No error

ScDoScan

The `ScDoScan` function allows your application to read scanned image data that has been collected by the driver. Your application's first call to `ScDoScan` starts a scan. To retrieve all the image data for a scan, your application should invoke this function repeatedly until it receives the `scEOS` result code, indicating that there is no more image data. To cancel a scan in progress, call the `ScAbortScan` function described earlier in this chapter.

During a scan on the Apple Scanner or the OneScanner, any call on any command other `ScDoScan` will also cause the scan to be aborted. During a scan on the Color OneScanner, only the `ScAbortScan` command will abort the scan. Any other command will return an `scBusy` result code.

Before beginning a scan, your application must set the parameters governing that scan using the `ScSetScanArea` function described later in this chapter.

PASCAL

```
FUNCTION ScDoScan (refNum: Integer; buffer: Ptr;VAR count:
LongInt; unused: Integer; byteWidth,rowBytes: Integer) : OSErr;
```

Assembly-Language Note

Calling `ScDoScan` is equivalent to calling the Device Manager `Read` routine with `ioParam` set as follows: `ioBuffer = buffer`, `ioReqCount = count`, `ioActCount = count`, `ioPosMode = unused`, high word of `ioPosOffset = byteWidth`, low word of `ioPosOffset = rowBytes`. ◆

`refNum` Scanner identifier returned by `ScOpen`.

`buffer` Pointer to the image buffer in the host computer's memory. The driver returns the scanned image data to this location.

`count` Maximum number of data bytes the driver is to read into the location specified by `buffer`. The driver updates this parameter with the count of bytes actually read.

Your application must observe two restrictions when setting this parameter. First, for smooth scanner operation, it must set `count` to a value at least as large as that stored in the `minReadSize` field of the `ScStdFeaturesRec` record returned by the `ScGetStdFeatures` function. Second, your application must set `count` to a multiple of `byteWidth` (described later), unless the program is reading the data into contiguous memory.

IMPORTANT

`Count` must not be less than one scan line. ▲

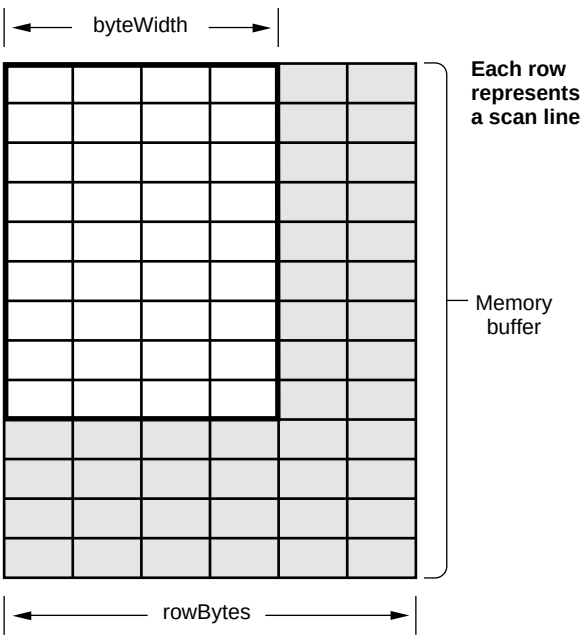
Note that the driver may return less data than requested. Thus your application should always check the returned value of `count` to find out how much data was actually transferred. The driver always sets `count` to a multiple of `byteWidth` unless your program is reading into contiguous memory. If the returned value in `count` is 0, the driver transferred no data because none was currently available. On the application's second call, to `ScDoScan`, the driver does not return to the application until data is available from the scanner. To prevent it from freezing while it waits for the scanner, the application may want to pause before issuing the second `ScDoScan` call.

`unused` Reserved for future expansion. Set this parameter to 0.

`byteWidth` Width of a scan line, in bytes. This parameter controls the number of bytes to be returned for each scan line. When reading image data into contiguous memory, your application should set this parameter to 0.

Used together, the `byteWidth` and `rowBytes` parameters allow your application to scan into noncontiguous memory, effectively placing a rectangle of scanned image data within a larger buffer (see Figure 3-1). The `byteWidth` parameter indicates the width of the new image data; the `rowBytes` parameter indicates the width of the receiving buffer. As the driver writes image data into the location specified by `buffer`, it writes rows that are `byteWidth` bytes long. At the end of a row, the driver moves to the start of the next row by skipping (`rowBytes`–`byteWidth`) bytes.

Figure 3-1 Scanning into noncontiguous memory



Each scan line must begin and end on a byte boundary. If the scan area for an image does not align with a byte boundary, you must adjust the value of the `byteWidth` parameter accordingly. Once you have scanned the image, your program should then mask off any unwanted pixels (see Figure 3-2, later in this chapter, for an example of a scan area that does not align with a byte boundary).

You can use the following formulas to calculate the value of the `byteWidth` parameter for a scan area (be sure to discard the remainders from each of the division operations):

- 4-bit and 8-bit scanners

  ([bits per pixel ∗ (scan rectangle right x – 1)] / 8) + 1 –
  ([bits per pixel ∗ scan area left x] / 8)

- Bi-level Color Composition mode

  (scan rectangle right x - Scan rectangle left x +7)/8∗3

- Remaining composition modes

  (((Scan rectangle right x - Scan rectangle left x) ∗ bits per pixel) + 7)/8

For example, if you are using an Apple Scanner or a OneScanner, and you want to scan an image bounded by a rectangle with left and right x coordinates of 10 and 200, respectively, in 16-level gray scale (which requires 4 bits per pixel), you would calculate the value of the `byteWidth` parameter as follows:

([4 ∗ (200 – 1)] / 8) + 1 – ([4 ∗ 10] / 8) = 95

`rowBytes` Width, in bytes, of the bitmap in the buffer, from the beginning of the bitmap to the next word boundary. This width must always be an even number of bytes. When reading image data into contiguous memory, your program should set this parameter to 0.

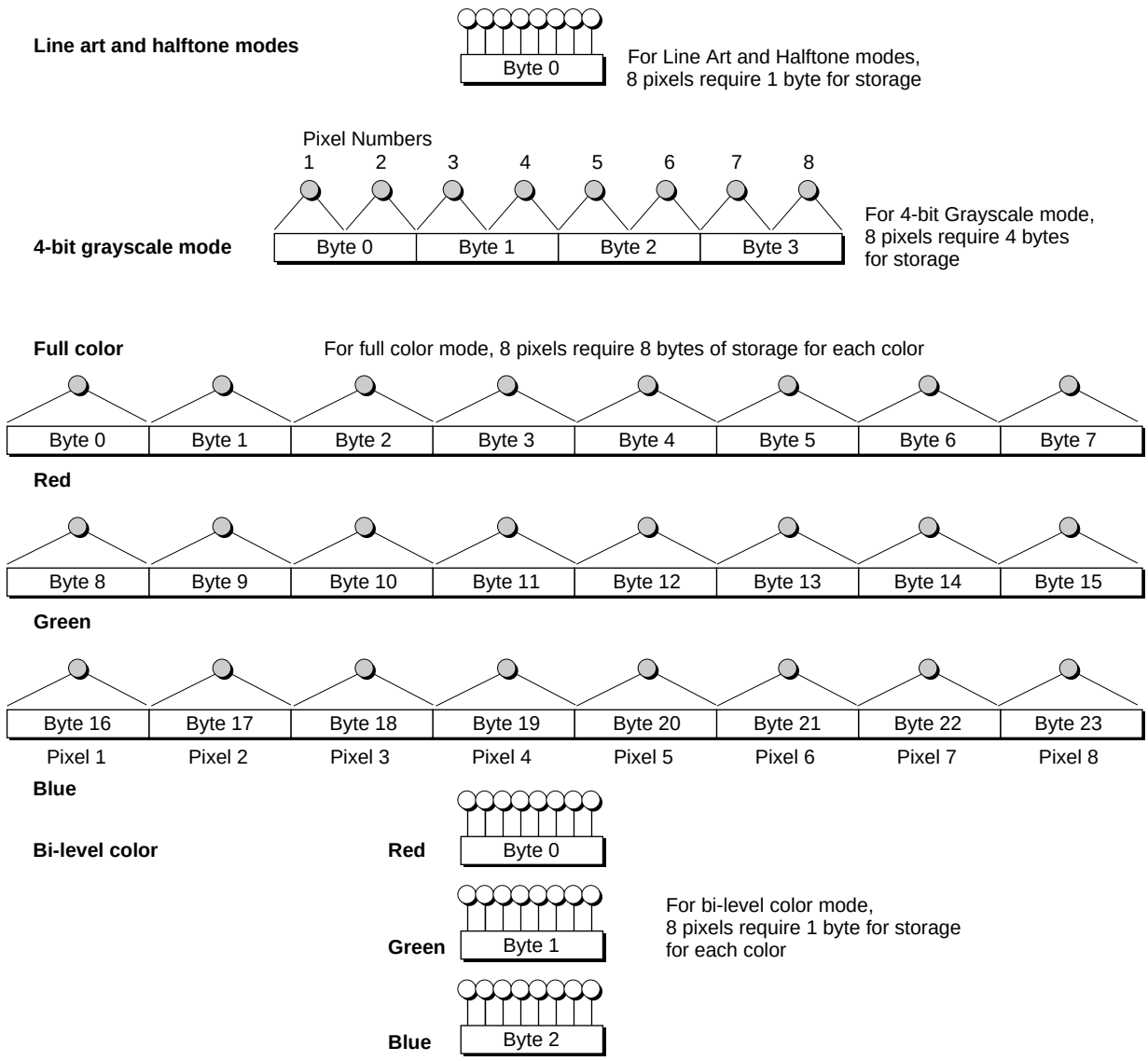
Note that the `PixMap` and `BitMap` records both contain fields named `rowBytes`. These fields define the width, in bytes, of the corresponding pixel map or bitmap, respectively. When reading scanned data into a pixel map or a bitmap, you may find it convenient to use the `rowBytes` field from the `PixMap` or `BitMap` record to supply the value for this parameter.

The values you calculate for `byteWidth` and `rowBytes` depend upon the composition mode you have selected. Eight pixels in Line Art mode or Halftone mode require 1 byte of memory, but 8 pixels in Grayscale mode require 4 bytes of memory (in 4-bit mode) or 8 bytes of memory (in 8-bit mode).

Full Color mode uses eight 8-bit pixels for each of the three colors (red, green, and blue). It therefore requires 24 bytes of memory. The 8 single-bit pixels used in Bi-level Color mode require 3 bytes of memory, one for each of the three colors.

Figure 3-2 shows how the composition mode affects these parameters.

Figure 3-2 How composition mode affects `rowBytes` and `byteWidth`



RESULT CODES

`noErr`	0	No error
`scComErr`	-17065	Communication-interface malfunction
`scResetErr`	-17066	Scanner reset or reinitialized
`scParamErr`	-17067	Illegal parameter or command
`scScannerErr`	-17068	Internal scanner malfunction
`scLampErr`	-17069	Lamp malfunction
`scEOS`	-17070	End of scan
`scDimLampErr`	-17071	Dim lamp and reduced scan quality (reissuing `ScDoScan` causes scanning to continue and the error to be suppressed)
`scBusy`	-17072	Driver call made while scanner is busy (Color OneScanner only)

ScGetHalfTones

The `ScGetHalfTones` function returns a list of the halftone patterns that the attached scanner supports for a specified composition mode. Your application can then present the list and allow the user to select the pattern for a given scanned image. The `ScGetHalfTones` function returns the pattern information in an `ScHalfToneArray` structure, described in Chapter 4.

The size of the returned `ScHalfToneArray` structure varies according to the number of halftone patterns supported by the attached scanner in the specified composition mode. Consequently, your application should use the Memory Manager to allocate the memory for this structure. Your application can determine the number of supported halftone patterns by issuing the `ScGetStdFeatures` function, described in Chapter 4.

When your application requests a scan in Halftone mode, it must specify the halftone pattern for the scan. You identify the halftone pattern name by supplying its index in the returned array of supported patterns in the `halfTone` field of the `ScAreaRec` record. (See "`ScAreaRec` in Chapter 4, and "`ScSetScanArea`," later in this chapter.)

PASCAL

```
FUNCTION ScGetHalfTones (refNum: Integer;compType:
Integer;halfTonePtr: ScHalfTonePtr) : OSErr;
```

Assembly-Language Note

`ScGetHalfTones` is equivalent to calling the Device Manager `Status` routine with `csCode = 4`, `csParam = compType`, and `csParam + 2 = halfTonePtr`. ◆

`refNum` Scanner identifier returned by `ScOpen`.

`compType` Composition mode for the request. Your application can determine the valid composition modes for the attached scanner by examining the `ScStdFeaturesRec` record returned by the `ScGetStdFeatures` function.

`halfTonePtr`
Pointer to the location where `ScGetHalfTones` is to return the resulting `ScHalfToneArray`. The `halfToneElements` field of the `ScCompRec` record corresponding to the composition mode specified by `compType` indicates the number of elements in the `ScHalfToneArray` structure formatted by `ScGetHalfTones`. The location specified by `halfTonePtr` must be able to receive an array of the appropriate size for that number of elements. See "`ScHalfToneArray`," in Chapter 4, for details on the format and content of this structure.

RESULT CODES

`noErr`	0	No error
`scParamErr`	-17067	Illegal parameter or command

ScGetRes

The `ScGetRes` function returns information about the resolutions supported by the attached scanner for a specified composition mode. Your application presents this information and allows the user to select a resolution value for an image to be scanned. The `ScGetRes` function returns the resolution information in an `ScResArray` structure described in Chapter 4.

The size of the returned `ScResArray` structure varies according to the number of resolutions supported by the attached scanner in the specified composition mode. Your application should use the Memory Manager to allocate the memory for this structure. Your application can determine the number of supported resolutions by issuing the `ScGetStdFeatures` function. (See "`ScGetStdFeatures`," later in this chapter.)

PASCAL

```
FUNCTION ScGetRes (refNum: Integer; compType: Integer;
resPtr: ScResPtr) : OSErr;
```

Assembly-Language Note

`ScGetRes` is equivalent to calling the Device Manager `Status` routine with `csCode = 3`, `csParam = compType`, and `csParam + 2 = resPtr`. ◆

`refNum` Scanner identifier returned by `ScOpen`.

`compType` Composition mode for the request. Your application can determine the valid composition modes for the attached scanner by examining the `ScStdFeaturesRec` record returned by the `ScGetStdFeatures` function.

`resPtr` Pointer to the location where `ScGetRes` is to return the resulting `ScResArray` structure. The `resElements` field of the `ScCompRec` record corresponding to the composition mode specified by `compType` indicates the number of elements in the `ScResArray` structure formatted by `ScGetRes`. The location specified by `resPtr` must be able to receive an array of the appropriate size for that number of elements. See "`ScResArray`," in Chapter 4, for details on the format and content of this structure.

If the attached scanner supports resolutions in increments of 1 dpi, the first and last elements in the resulting `ScResArray` structure represent the upper and lower resolution limits for the scanner. See the description of `resFlags` field of the `ScCompRec` record in "`ScCompRec`," in Chapter 4, for more information.

RESULT CODES

`noErr`	0	No error
`scParamErr`	-17067	Illegal parameter or command
`scBusy`	-17072	Driver call made while scanner is busy (Color OneScanner only)

ScGetStdFeatures

The `ScGetStdFeatures` function returns information about the capabilities of the attached scanner. Your application should use this returned information to restrict the options presented to the user and the values of the scanning control parameters. Your application sets these values by calling the `ScSetScanArea` function, described later in this chapter. The `ScGetStdFeatures` function returns the feature information in an `ScStdFeaturesRec` record, described in Chapter 4.

PASCAL

```
FUNCTION ScGetStdFeatures (refNum: Integer; stdFeaturesPtr:
ScStdFeaturesPtr; length: Integer) : OSErr;
```

Assembly-Language Note

`ScGetStdFeatures` is equivalent to calling the Device Manager `Status` routine with `csCode = 2`, `csParam = stdFeaturesPtr`, and `csParam + 4 = length`. ◆

`refNum` Scanner identifier returned by `ScOpen`.

`stdFeaturesPtr`
A pointer to the location where the `ScGetStdFeatures` function is to return a formatted `ScStdFeaturesRec` record. The `length` parameter indicates the size of this receiving buffer. See "`ScStdFeaturesRec`" in Chapter 4, for details on the format and content of this structure.

`length` Size, in bytes, of the receiving buffer specified by the `stdFeaturesPtr` parameter. The driver limits the structure to this size so that, in future releases, new information may be added to the `ScStdFeaturesRec` record without causing existing applications to crash. If your application asks for more data than is available, the `ScGetStdFeatures` function returns only the available data. By checking the `version` field in the `ScStdFeaturesRec` record, your application can determine the size of the returned structure.

RESULT CODES

`noErr`	0	No error
`scParamErr`	-17067	Illegal parameter or command
`scBusy`	-17072	Driver call made while scanner is busy (Color OneScanner only)

ScOpen

The `ScOpen` function opens the scanner driver and prepares the driver to receive commands. After invoking a call to `ScOpen`, the parameter `refNum` contains the identification number of the scanner. This value must be provided in all other calls to the driver during this session. Your application ends the scanner session by calling the `ScClose` function, described earlier in this chapter. Only one application at a time may have the scanner driver open.

The device driver is named .Scanner, and it is a standard Macintosh driver.

PASCAL

```
FUNCTION ScOpen (VAR refNum: Integer) : OSErr;
```

Assembly-Language Note

Calling `ScOpen` is equivalent to calling the Device Manager `Open` routine, except that `ScOpen` returns an error if another program has already opened the scanner driver. ◆

`refNum` Buffer that receives the reference number for use with all other scanner driver functions in this session.

RESULT CODES

`noErr`	0	No error
`badUnitErr`	-21	Scanner is not connected or, if connected, is not switched on
`openErr`	-23	Another program has opened the scanner driver
`scNotFoundErr`	-17064	Scanner not found
`scScannerErr`	-17068	Internal scanner malfunction
`scLampErr`	-17069	Lamp is too dim to operate correctly or CCD is not functioning properly
`scBusy`	-17072	Driver call made while scanner is busy (Color OneScanner only)

ScSetScanArea

The `ScSetScanArea` function establishes the parameters that control a scan. Before your application can begin a scan, it must set such scanning parameters as resolution, scan area, brightness, contrast, composition, and number of bits per pixel of the area to be scanned. Your application sets these control parameters by formatting an `ScScanAreaRec` record and then passing to the `ScSetScanArea` function a pointer to the area formatted. Your application should determine the valid values for these parameters by issuing the `ScGetStdFeatures` driver function before formatting the `ScScanAreaRec` record for this routine. (See the description of `ScGetStdFeatures`, earlier in this chapter.)

PASCAL

```
FUNCTION ScSetScanArea (refNum: Integer; scanAreaPtr:
ScScanAreaPtr) : OSErr;
```

Assembly-Language Note

`ScSetScanArea` is equivalent to calling the Device Manager Control routine with `csCode = 2` and `csParam = scanAreaPtr`. ◆

`refNum` Scanner identifier returned by `ScOpen`.

`scanAreaPtr`
Pointer to a formatted `ScScanAreaRec` record. See the description of `ScScanAreaRec`, in Chapter 4, for details on the format and content of the structure.

RESULT CODES

`noErr`	0	No error
`scComErr`	-17065	Communication-interface malfunction
`scParamErr`	-17067	Illegal parameter or command
`scResetErr`	-17066	Scanner reset or reinitialized
`scScannerErr`	-17968	Internal scanner malfunction
`scBusy`	-17072	Driver call made while scanner is busy (Color OneScanner only)

Advanced Functions

The following sections describe the advanced scanner driver functions that supplement the standard driver functions and allow applications to take advantage of special features of the attached scanner. Most applications do not need to use these functions.

Note
Many parameters sent to the driver, or returned from the driver, are listed as reserved. These parameters must be set to 0 if they are supplied to the driver. If they are parameters returned from the driver, they will always contain a value of 0. ◆

Advanced functions that apply to all scanners:

- `ScGetAdvFeatures`
- `ScSetLamp`
- `ScSetLED`
- `ScVendorUnique`

Advanced functions for the Apple Scanner:

- `ScSetGrayMap`
- `ScSetGroup3`
- `ScSetHTPattern`
- `ScSetNoHome`
- `ScSetThreshold`
- `ScWaitButton`

Advanced functions for the OneScanner:

- `ScGetButton` (available ROM version 2.03 or earlier)
- `ScResetButton` (available only with ROM version 2.03, or earlier)
- `ScSetHTPattern`
- `ScSetNoCal`
- `ScSetSpeed`

Advanced functions for the Color OneScanner:

- `ScInvertPixels`
- `ScLoadGamma`
- `ScLoadMatrix`
- `ScSensorSelect`
- `ScSetScannerAtoD`

ScGetAdvFeatures

The `ScGetAdvFeatures` function identifies which advanced driver features the attached scanner supports. If your application needs to use any of the advanced driver features, it should first use the `ScGetAdvFeatures` function to determine the advanced capabilities of the attached scanner. This function returns the advanced feature information in an `ScAdvFeaturesRec` record. (See "`ScAdvFeaturesRec`," in Chapter 4, for information on the format and content of that structure.)

Your application invokes individual features by calling the appropriate driver functions. A portion of the returned `ScAdvFeaturesRec` record indicates which of these functions are supported by the attached scanner. Subsequent sections of this chapter discuss each function in detail.

PASCAL

```
FUNCTION ScGetAdvFeatures (refNum: Integer; advFeaturesPtr:
ScAdvFeaturesPtr; length: Integer) : OSErr;
```

Assembly-Language Note

Calling `ScGetAdvFeatures` is equivalent to calling the Device Manager `Status` routine with `csCode = 5`, `csParam = advFeaturesPtr`, and `csParam + 4 = length`. ◆

`refNum` Scanner identifier returned by `ScOpen`.

`advFeaturesPtr`
Pointer to the location where the `ScGetAdvFeatures` function is to return a formatted `ScAdvFeaturesRec` record. The `length` parameter indicates the size of this receiving buffer. See "`ScAdvFeaturesRec`," later in this chapter, for details on the format and content of this structure.

`length` Size, in bytes, of the receiving buffer identified by the `advFeaturesPtr` parameter. The driver limits the structure to this size, so that in future releases, new information may be added to the `ScAdvFeaturesRec` record without causing existing applications to crash. If your application asks for more data than is available, the `ScGetAdvFeatures` function returns only the available data. By checking the `version` field in the `ScAdvFeaturesRec` record, your application can determine the size of the returned structure.

RESULT CODES

`noErr`	0	No error
`scParamErr`	-17067	Illegal parameter or command
`scBusy`	-17072	Driver call made while scanner is busy (Color OneScanner only)

ScGetButton

The `ScGetButton` function reads the button state maintained by the scanner driver. The scanner driver uses a Boolean value to track the state of the scanner button. When the user presses the scanner button, the driver sets the value to `TRUE`. The `ScGetButton` function allows your application to read the button state.

Your application can set the button state by calling the `ScResetButton` function, described later in this chapter.

IMPORTANT

`ScGetButton` is supported only on OneScanners with ROM version 2.03 and earlier. ▲

PASCAL

```
FUNCTION ScGetButton (refNum: Integer; VAR button: Boolean) :
OSErr;
```

Assembly-Language Note

Calling `ScGetButton` is equivalent to calling the Device Manager `Control` routine with `csCode = 6` and `csParam =` pointer to `button`. ◆

`refNum` Scanner identifier returned by `ScOpen`.

`button` Boolean value set by the scanner driver to match the stored button state. If the user pressed the button after the last call to the `ScResetButton` function, the driver sets this field to `TRUE`. Otherwise, the field is `FALSE`.

RESULT CODES

`noErr`	0	No error
`scComErr`	-17065	Communication-interface malfunction
`scResetErr`	-17066	Scanner reset or reinitialized
`scParamErr`	-17067	Illegal parameter or command
`scScannerErr`	-17068	Internal scanner malfunction

ScInvertPixels

The `ScInvertPixels` function inverts the values of full intensity pixels and black pixels. In RGB-direct mode 8, a value of 255 is full intensity, and a value of 0 is black. In other QuickDraw modes, the Macintosh computer, using 8- or 4-bit pixels provides a default grayscale color table where 0 is full intensity and 255 is black. A call to `ScInvertPixels` with `InvertFlag` set to true, causes 255 to equal black, and 0 to equal full intensity. A call to `ScSetScanArea` resets the `ScInvertPixels` bit to produce an image that is normal or not inverted.

Note
On the Color OneScanner, `ScInvertPixels` is set to `TRUE` by the driver when gray or Line Art composition modes are selected in `ScSetScanArea`. Calls to `ScSetScanArea` in other modes set `ScInvertPixels` to `FALSE`. ◆

PASCAL

```
FUNCTION ScInvertPixels (refNum: Integer; InvertFlag : BOOLEAN)
: OSErr;
```

Assembly-Language Note
Calling `ScInvertPixels` is equivalent to a Control call with `csCode` = 17 and `csParam = InvertFlag`. ◆

`refNum` Scanner identifier returned by `ScOpen`.

`InvertFlag`
A Boolean value that should be set to `FALSE` in Grayscale mode, to invert the image. In Bi-level or Full Color mode it should be set to `TRUE` to get a negative image.

RESULT CODES

`noErr`	0	No error
`scComErr`	-17065	Communication-interface malfunction
`scResetErr`	-17066	Scanner reset or reinitialized
`scParamErr`	-17067	Illegal parameter or command
`scScannerErr`	-17068	Internal scanner malfunction
`scBusy`	-17072	Driver call made while scanner is busy

ScLoadGamma

The `ScLoadGamma` function loads a table of values used to change the scanner's intensity curve. Each scanned pixel value is used as an address for this table, and the bytes from the table (R, G, and B) at that address are returned as the pixel's intensity.

PASCAL

```
FUNCTION ScLoadGamma (refNum: Integer; GammaTable:
scGammaTablePtr): OSErr;

scGammaTablePtr = ^ scGammaTableRec;
scGammaTableRec = PACKED RECORD
                  RedGamma : ARRAY[1..256] OF BYTE;
                  GreenGamma : ARRAY[1..256] OFBYTE;
                  BlueGamma : ARRAY[1..256] OF BYTE;
```

Assembly-Language Note

Calling `ScGamma` is equivalent to calling the Device Manager with `csCode` = 19 and `csParam = scGammaTablePtr`. ◆

`refNum` Scanner identifier returned by `ScOpen`.

`Table:scGammaTablePtr`
Defines a pointer to an `ScGammaTablePtr` record.

RESULT CODES

`noErr`	0	No error
`scComErr`	-17065	Communication-interface malfunction
`scResetErr`	-17066	Scanner reset or reinitialized
`scParamErr`	-17067	Illegal parameter or command
`scScannerErr`	-17068	Internal scanner malfunction
`scBusy`	-17072	Driver call made while scanner is busy

ScLoadMatrix

The `ScLoadMatrix` function loads the matrix multiplier with the array data. The Color OneScanner multiplier is 16 bits. Data should always be left justified, and loaded in row order as follows:

1 2 3
4 5 6
7 8 9

A call to `ScSetScanArea` loads the matrix with a set of default values that are specific to the composition mode indicated in the `ScSetScanArea` call. In Grayscale mode, `ScLoadMatrix` should be called after an `ScSetScanArea` call, so that the new matrix values may take effect.

PASCAL

```
FUNCTION ScLoadMatrix (refNum: Integer; Matrix:scMatrixPtr:)
OSErr;
ScMatrixPtr=^scMatrix;
ScMatrix; array [1..9] OF INTEGER
```

Assembly-Language Note

Calling `ScLoadMatrix` is equivalent to calling the Device Manager with `csCode` = 18 and `csParam = scMatrixPtr(Ptr)`. ◆

`refNum` Scanner identifier returned by `ScOpen`.

`scMatrixPtr`
Defines a pointer to an `ScMatrix` array.

RESULT CODES

`noErr`	0	No error
`scComErr`	-17065	Communication-interface malfunction
`scResetErr`	-17066	Scanner reset or reinitialized
`scParamErr`	-17067	Illegal parameter or command
`scScannerErr`	-17068	Internal scanner malfunction
`scBusy`	-17072	Driver call made while scanner is busy

ScResetButton

The `ScResetButton` function resets the button state maintained by the scanner driver. The scanner driver uses a Boolean value to track the state of the scanner button. When the user presses the scanner button, the driver sets that value to `TRUE`. The `ScResetButton` function allows your application to set this value to `FALSE` so that it can detect subsequent button presses.

Your application can read the button state by calling the `ScGetButton` function, described earlier in this chapter.

IMPORTANT

`ScResetButton` is supported only on OneScanners with ROM version 2.03 and earlier. ▲

PASCAL

```
FUNCTION ScResetButton (refNum: Integer; setTrue: Boolean): OSErr;
```

Assembly-Language Note

Calling `ScResetButton` is equivalent to calling the Device Manager Control routine with `csCode = 13` and `csParam = setTrue`. ◆

`refNum` Scanner identifier returned by `ScOpen`.

`setTrue` Control for the button state. Set this parameter to `TRUE` if you want the driver to reset the button state to `FALSE`.

RESULT CODES

`noErr`	0	No error
`scComErr`	-17065	Communication-interface malfunction
`scResetErr`	-17066	Scanner reset or reinitialized
`scParamErr`	-17067	Illegal parameter or command
`scScannerErr`	-17068	Internal scanner malfunction

ScSensorSelect

The `ScSensorSelect` function allows an application to select which sensor is used for scanning gray on a color scanner.

PASCAL

```
FUNCTION ScSensorSelect (refNum: Integer;sensor:Integer) : OSErr;
```

Assembly-Language Note

Calling `ScSensorSelect` is equivalent to calling the Device Manager with `csCode = 16` and `csParam = sensor`. ◆

`refNum` Scanner identifier returned by `ScOpen`.

`sensor`

`scAllSensors`	= 0x00
`scRedSensor`	= 0x01
`scGreenSensor`	= 0x02
`scBlueSensor`	= 0x03

RESULT CODES

`noErr`	0	No error
`scComErr`	-17065	Communication-interface malfunction
`scResetErr`	-17066	Scanner reset or reinitialized
`scParamErr`	-17067	Illegal parameter or command
`scScannerErr`	-17068	Internal scanner malfunction
`scBusy`	-17072	Driver call made while scanner is busy

ScSetGraymap

The `ScSetGraymap` function sets the graymap curve used during scanning. By changing the graymap curve used for a scan, your application can enhance the visual detail in either the light or dark portions of the scanned image. In this manner, you can bring forth detail that would normally be lost to human sight. For a description of this phenomenon, see "Gamma Correction" in Chapter 1, "A Scanning Primer."

PASCAL

```
FUNCTION ScSetGraymap (refNum: Integer; grayMap: Integer) : OSErr;
```

Assembly-Language Note

Calling `ScSetGraymap` is equivalent to calling the Device Manager Control routine with `csCode = 8` and `csParam = grayMap`. ◆

`refNum` Scanner identifier returned by `ScOpen`.

`grayMap` Indicator of whether the scanner should try to enhance visual details in the scanned image and, if so, the technique to use. The scanner driver supports the following three values for this parameter:

`scLightDetail`
Scanner driver artificially enhances detail in the lighter portions of the scanned image.

`scNormalDetail`
Scanner driver does not enhance the image in any way, but represents the scanned image as perceived by the human eye.

`scDarkDetail`
Scanner driver artificially enhances detail in the darker portions of the scanned image.

RESULT CODES

`noErr`	0	No error
`scComErr`	-17065	Communication-interface malfunction
`scResetErr`	-17066	Scanner reset or reinitialized
`scParamErr`	-17067	Illegal parameter or command
`scScannerErr`	-17068	Internal scanner malfunction

ScSetGroup3

The `ScSetGroup3` function controls the Group III compression feature of the scanner driver. The Group III compression feature allows the scanner to represent data using standard FAX encoding, to meet the CCITT (Consultative Committee on International Telegraphy and Telephony), Group III standard. If your application enables Group III encoding, the scanner driver uses subsequent calls to the `ScSetScanArea` function to define areas to be scanned with Group III, one-dimensional encoding. The driver returns data in Group III format until you turn this feature off. When your program reads compressed data, it must set the `byteWidth` and `rowBytes` parameters of the `ScDoScan` function to 0.

PASCAL

```
FUNCTION ScSetGroup3 (refNum: Integer; compressOn: Boolean) :
OSErr;
```

Assembly-Language Note

Calling `ScSetGroup3` is equivalent to calling the Device Manager Control routine with `csCode = 5` and `csParam = compressOn`. ◆

`refNum` Scanner identifier returned by `ScOpen`.

`compressOn` Control for Group III encoding. Set this parameter to `TRUE` to enable Group III encoding for subsequent scans; set it to `FALSE` to disable Group III encoding.

RESULT CODES

`noErr`	0	No error
`scComErr`	-17065	Communication-interface malfunction
`scResetErr`	-17066	Scanner reset or reinitialized
`scParamErr`	-17067	Illegal parameter or command
`scScannerErr`	-17068	Internal scanner malfunction

ScSetHTPattern

The `ScSetHTPattern` function causes the scanner driver to download a custom halftone pattern into the attached scanner. A halftone pattern consists of a matrix of brightness values. In Halftone mode, the scanner filters scanned data through that matrix, turning result pixels on or off according to the brightness of the scanned pixel relative to the appropriate matrix value. The resulting two-level data emulates gray-scale encoding in much the same way that dithering a color image fools the eye into seeing colors that are not there.

Your application formats the halftone matrix into an `ScPatRec` record, described in Chapter 4. To select the downloaded pattern for a particular scan, your application must assign a value of -1 to the `halfTone` field, in the `ScAreaRec` record passed to the `ScSetScanArea` function. (See the description of `ScSetScanArea`, earlier in this chapter.)

PASCAL

```
FUNCTION ScSetHTPattern (refNum: Integer; patPtr: ScPatPtr) :
OSErr;
```

Assembly-Language Note

Calling `ScSetHTPattern` is equivalent to calling the Device Manager `Control` routine with `csCode = 4` and `csParam = patPtr`. ◆

`refNum` Scanner identifier returned by `ScOpen`.

`patPtr` Pointer to an `ScPatRec` record that defines the halftone pattern to be loaded into the attached scanner. See the description of `ScPatRec`, Chapter 4, for information on the format and content of this record.

RESULT CODES

`noErr`	0	No error
`scComErr`	-17065	Communication-interface malfunction
`scResetErr`	-17066	Scanner reset or reinitialized
`scParamErr`	-17067	Illegal parameter or command
`scScannerErr`	-17068	Internal scanner malfunction

ScSetLamp

The `ScSetLamp` function turns the scanner lamp on and off. Turning on the lamp warms it up in advance of the scan.

PASCAL

```
FUNCTION ScSetLamp (refNum: Integer; lampOn: Boolean) : OSErr;
```

Assembly-Language Note

Calling `ScSetLamp` is equivalent to calling the Device Manager `Control` routine with `csCode = 7` and `csParam = lampOn`. ◆

`refNum` Scanner identifier returned by `ScOpen`.

`lampOn` Control for the lamp. A value of `TRUE` turns the lamp on. The scanner driver leaves the lamp on for two minutes after the last SCSI access, and then turns the lamp off. Each subsequent SCSI access causes the driver to turn the lamp on for two more minutes. A value of `FALSE` turns the lamp off. Because the lamp does not turn off automatically when you close the driver, remember to turn it off before calling the `ScClose` function to close the driver.

RESULT CODES

`noErr`	0	No error
`scComErr`	-17065	Communication-interface malfunction
`scResetErr`	-17066	Scanner reset or reinitialized
`scParamErr`	-17067	Illegal parameter or command
`scScannerErr`	-17068	Internal scanner malfunction
`scBusy`	-17072	Driver call made while scanner is busy

ScSetLED

The `ScSetLED` function controls the setting of the scanner light-emitting diode (LED).

PASCAL

```
FUNCTION ScSetLED (refNum: Integer; LEDOn: Boolean) : OSErr;
```

Assembly-Language Note

Calling `ScSetLED` is equivalent to calling the Device Manager `Control` routine with `csCode = 12` and `csParam = LEDOn`. ◆

`refNum` Scanner identifier returned by `ScOpen`.

`ledOn` Control for the scanner LED. Set this parameter to `TRUE` if you want the scanner to turn the LED on. Set this parameter to `FALSE` to turn the LED off.

RESULT CODES

`noErr`	0	No error
`scComErr`	-17065	Communication-interface malfunction
`scResetErr`	-17066	Scanner reset or reinitialized
`scParamErr`	-17067	Illegal parameter or command
`scScannerErr`	-17068	Internal scanner malfunction
`scBusy`	-17072	Driver call made while scanner is busy (Color OneScanner only)

ScSetNoCal

The `ScSetNoCal` function controls whether the scanner calibrates itself for lamp intensity before scanning. The fluorescent lamp in the scanner loses intensity as it ages. By default, the scanner calibrates for lamp intensity, so that it can compensate for the lower light output of an older lamp. This calibration step takes a few seconds. `ScSetNoCal` allows your application to skip this calibration step, thus reducing scan time. However, skipping lamp calibration also compromises scan quality, and your application should turn off calibration only for preview scans.

PASCAL

```
FUNCTION ScSetNoCal (refNum: Integer; noCalMode: Boolean) : OSErr;
```

Assembly-Language Note

Calling `ScSetNoCal` is equivalent to calling the Device Manager `Control` routine with `csCode = 14` and `csParam = noCalMode`. ◆

`refNum` Scanner identifier returned by `ScOpen`.

`noCalMode` Indicator of whether the scanner should calibrate the scan for the current lamp intensity before beginning subsequent scans. Set this parameter to `TRUE` to force lamp calibration before each scan. Set this parameter to `FALSE` to skip lamp calibration.

RESULT CODES

`noErr`	0	No error
`scComErr`	-17065	Communication-interface malfunction
`scResetErr`	-17066	Scanner reset or reinitialized
`scParamErr`	-17067	Illegal parameter or command
`scScannerErr`	-17068	Internal scanner malfunction

ScSetNoHome

The `ScSetNoHome` function controls whether the carriage assembly returns to the home position after the scanner has finished a scan. By default, the scanner driver configures the attached scanner to return the carriage to the home position after each scan. However, this default may result in a great deal of carriage movement when your application conducts a series of complex scans. Your application can speed up complex scanning operations by using this function to override that default. The carriage assembly then remains where it was at the end of the scan.

In addition, your application can move the carriage assembly to a particular position by issuing the `ScSetNoHome` function to override the homing action and then scanning an area of infinitely small size (with the `ScDoScan` function).

PASCAL

```
FUNCTION ScSetNoHome (refNum: Integer; noHome: Boolean) : OSErr;
```

Assembly-Language Note

Calling `ScSetNoHome` is equivalent to calling the Device Manager `Control` routine with `csCode = 6` and `csParam = noHome`. ◆

`refNum` Scanner identifier returned by `ScOpen`.

`noHome` Control for the carriage assembly after a scan. Set this parameter to `TRUE` to prevent the carriage assembly from returning to the home position. Set it to `FALSE` to force the carriage home after each scan.

RESULT CODES

`noErr`	0	No error
`scComErr`	-17065	Communication-interface malfunction
`scResetErr`	-17066	Scanner reset or reinitialized
`scParamErr`	-17067	Illegal parameter or command
`scScannerErr`	-17068	Internal scanner malfunction

ScSetScannerAtoD

The `ScSetScannerAtoD` function gives your application direct access to the analog-to-digital converter used in the scanning process. It enables the application to control brightness and contrast, using `Vrt` and `Vrb`, which override any other brightness and contrast controls. The call to `ScSetScannerAtoD` should be made before the call to `ScSetScanArea`. This ensures that the settings specified by `ScSetScannerAtoD` take effect.

Note
This call is valid for the Color OneScanner only, for grayscale and color composition modes. ◆

PASCAL

```
FUNCTION ScSetScannerAtoD (refNum: Integer; Vrt, Vrb :BYTE):
OSErr;
```

Assembly-Language Note
Calling `ScSetScannerAtoD` is equivalent to calling the Device Manager `Control` routine with `csCode = 17` and `csParam = Vrt, csParam+2 = Vrb`. ◆

`refNum` Scanner identifier returned by `ScOpen`.

`Vrt`, `Vrb`: Use `Vrt` to specify the top reference voltage for the ADC (analog-to-digital converter). Use `Vrb` to specify the bottom reference voltage for the ADC. A value of 0 in either `Vrt` or `Vrb` indicates that the Brightness and Contrast should be used instead, `Vrt` and `Vrb` are disabled. A value of 1 through 255 indicates the scanner should use a relative value. The default of 255 covers the full dynamic range.

RESULT CODES

`noErr`	0	No error
`scComErr`	-17065	Communication-interface malfunction
`scResetErr`	-17066	Scanner reset or reinitialized
`scParamErr`	-17067	Illegal parameter or command
`scScannerErr`	-17068	Internal scanner malfunction
`scBusy`	-17072	Driver call made while scanner is busy

ScSetSpeed

The `ScSetSpeed` function controls the speed of data transfer between the scanner driver and the attached scanner. The scanner driver supports three transfer speeds: normal, high-speed, and fast. Normal and high-speed transfers guarantee data integrity no matter how often your application calls `ScDoScan` to read scanned ***data.*** However, the transfer protocol between the scanner and scanner driver introduces some overhead. Fast transfers eliminate this protocol overhead and thus substantially reduce data transfer time. By eliminating the protocol overhead, however, fast transfers also eliminate the guarantee of data integrity.

Consequently, your application should use fast transfers only when you can be sure that your program can absorb all the scanned image data. In general, you should use fast transfer only with high-performance CPUs and only when sufficient memory is available to store the entire scanned image. The scanner driver supports only normal speed transfers on low-performance systems.

Note
If interrupts occur during a scan operation in fast transfer speed, the scan may be affected. ◆

PASCAL

```
FUNCTION ScSetSpeed (refNum: Integer; speed: Integer) : OSErr;
```

Assembly-Language Note
`ScSetSpeed` is equivalent to calling the Device Manager `Control` routine with `csCode = 11` and `csParam = speed`. ◆

`refNum` Scanner identifier returned by `ScOpen`.

`speed` Data transfer speed for subsequent scanner operations. The scanner driver supports the following three values for this parameter:

`scSpeedNormal`
Selects normal transfer. Data integrity is guaranteed.

`scSpeedHigh`
Selects high-speed transfer. Data integrity is guaranteed.

`scSpeedFast`
Selects fast transfer. Data integrity is guaranteed.

RESULT CODES

`noErr`	0	No error
`scComErr`	-17065	Communication-interface malfunction
`scResetErr`	-17066	Scanner reset or reinitialized
`scParamErr`	-17067	Illegal parameter or command
`scScannerErr`	-17068	Internal scanner malfunction

ScSetThreshold

The `ScSetThreshold` function sets the threshold level that the scanner uses with automatic background adjustment to determine which ***dots*** are black and which are white when scanning in Line Art mode.

PASCAL

```
FUNCTION ScSetThreshold (refNum: Integer; threshold: Integer) :
OSErr;
```

Assembly-Language Note

Calling `ScSetThreshold` is equivalent to calling the Device Manager `Control` routine with `csCode = 9` and `csParam = threshold`. ◆

`refNum` Scanner identifier returned by `ScOpen`.

`threshold` Values range from 1 to the value of the `brightnessMax` field in the `ScCompRec` record for the appropriate composition mode. (See the description of `ScGetStdFeatures` earlier in this chapter, and in Chapter 4.) A value of 0 selects a scanner-dependent default.

RESULT CODES

`noErr`	0	No error
`scComErr`	-17065	Communication-interface malfunction
`scResetErr`	-17066	Scanner reset or reinitialized
`scParamErr`	-17067	Illegal parameter or command
`scScannerErr`	-17068	Internal scanner malfunction

ScSetWaitButton

The `ScSetWaitButton` function controls whether the scanner driver waits for the user to press the scanner button before beginning a scan. If your application enables this feature and then calls `ScDoScan` to begin a scan, the driver does not actually start the scan operation until the user presses the scanner button. You can use this feature to ensure that the user has placed the image on the scanner glass before scanning begins.

PASCAL

```
FUNCTION ScSetWaitButton (refNum: Integer; waitButton: Boolean)
: OSErr;
```

Assembly-Language Note

Calling `ScSetWaitButton` is equivalent to calling the Device Manager `Control` routine with `csCode = 10` and `csParam = waitButton`. ◆

`refNum` Scanner identifier returned by `ScOpen`.

`waitButton` Control that determines whether the scanner driver waits for the user to press the scanner button before beginning a scan. If you set this parameter to `TRUE`, the driver scanner waits for the user to push the button. A value of `FALSE` turns this feature off.

RESULT CODES

`noErr`	0	No error
`scComErr`	-17065	Communication-interface malfunction
`scResetErr`	-17066	Scanner reset or reinitialized
`scParamErr`	-17067	Illegal parameter or command
`scScannerErr`	-17068	Internal scanner malfunction

ScVendorUnique

The `ScVendorUnique` function allows your application to invoke scanner functions that are unique to a particular scanner. This function provides a mechanism for passing a parameter type and corresponding parameter data to the scanner.

PASCAL

```
FUNCTION ScVendorUnique (refNum: Integer; paramType: Integer;
paramPtr: Ptr) : OSErr;
```

Assembly-Language Note

Calling `ScVendorUnique` is equivalent to calling the Device Manager `Control` routine with `csCode = 8192`, `csParam = paramType`, and `csParam + 2 = paramPtr`. ◆

`refNum` Scanner identifier returned by `ScOpen`.

`paramType` Parameter type. The following values are valid:

`scUniqueUnpark`
Instructs the scanner to move the carriage to the home position. The `paramPtr` parameter is not used.

`scUniquePark`
Instructs the scanner to move the carriage to the shipment lock position. The `paramPtr` parameter is not used.

`scUniqueAbsPos`
Instructs the scanner to position the carriage at the scan line specified in the `paramPtr` field. A value of 0 causes the scanner to place the carriage at the beginning of the scan area. Other values must indicate a valid y-axis position in increments of $^{1}/_{1200}$ of an inch.

`scUniqueRelPos`
Instructs the scanner to move the carriage to a position relative to its current position. The value of the `paramPtr` field indicates the direction and distance of the movement. Positive values move the carriage forward; negative values move the carriage backwards. The distance is expressed in increments of $^{1}/_{1200}$ of an inch. A value of 0 does not move the carriage.

`scUniqueSetCRAM`
Sets the contents of the scanner's calibration RAM. The `paramPtr` parameter must contain a pointer to a contiguous buffer of 2550 bytes. Each byte of parameter data contains the new calibration data for the corresponding pixel sensor in the CCD array.

`scUniqueGetCRAM`
Retrieves the contents of the scanner's calibration RAM. The `paramPtr` parameter must contain a pointer to a contiguous buffer of 2550 bytes. Each returned byte contains the calibration data for the corresponding pixel sensor in the CCD array.

`paramPtr` Parameter data appropriate to the parameter type specified in `paramType`.

RESULT CODES

`noErr`	0	No error
`scComErr`	-17065	Communication-interface malfunction
`scResetErr`	-17066	Scanner reset or reinitialized
`scParamErr`	-17067	Illegal parameter or command
`scScannerErr`	-17068	Internal scanner malfunction
`scBusy`	-17072	Driver call made while scanner is busy

CHAPTER 4

Scanner Driver Data Structures

Your application uses scanner driver data structures to communicate with the scanner driver. This chapter describes the format and content of two classes of data structures: standard and advanced. Standard data structures are used with standard driver functions, and advanced data structures are used with advanced driver functions.

Standard Data Structures

This section describes the data structures your application uses to interact with standard scanner driver functions. (See "Standard Functions," in Chapter 3, "Scanner Driver Functions.")

The structures described in this section are:

- `ScAreaRec`
- `ScCompRec`
- `ScHalfToneArray`
- `ScResArray`
- `ScScanAreaRec`
- `ScStdFeaturesRec`

ScAreaRec

The `ScAreaRec` record contains information defining a region to be scanned. Your application formats one `ScAreaRec` record for each area to be scanned in an operation, then passes those records to the `ScSetScanArea` driver function in an `ScScanAreaRec` record. (See the description of `ScSetScanArea`, in Chapter 3, "Scanner Driver Functions.")

Each `ScAreaRec` record specifies the scanning parameters for an area, including brightness, contrast, and composition mode, as well as the coordinates defining the scan area itself. These parameter values must be valid in the context of the scanner capabilities defined in the `ScStdFeaturesRec` record returned by the `ScGetStdFeatures` driver function, described in Chapter 3.

The following Pascal code shows the layout of the `ScAreaRec` record.

```
TYPE  ScAreaRec = RECORD
         reserved : LongInt;
         xDpi : Integer;
         yDpi : Integer;
         scanRect : Rect;
         brightness : Integer;
         contrast : Integer;
         composition : SignedByte;
         bitsPerPixel : SignedByte;
         halfTone : Integer;
      END;
```

The fields are defined as follows:

`reserved` This field is reserved for future expansion. Set this field to 0.

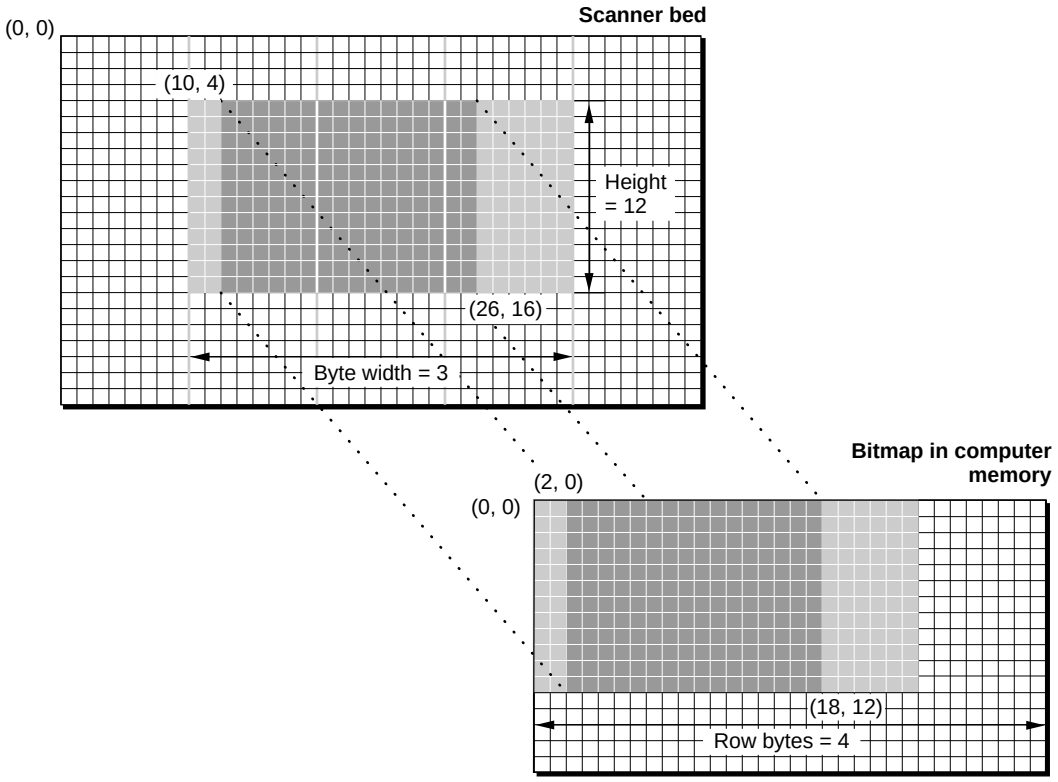
`xDpi` This field indicates the resolution in pixels per inch in the horizontal direction (along the x-axis of the document).

`yDpi` This field indicates the resolution in pixels per inch in the vertical direction (along the y-axis of the document).

The `xDpi` and `yDpi` fields specify the resolution at which the image is acquired. To scan the image without distortion, assign equal values to each of these two fields. You can achieve special effects by setting these parameters to different values.

`scanRect` This field specifies the size and the position of the scan area. The upper-right corner of the scanner glass as you look down at the scanner (which corresponds to the upper-left corner of the original document) is located at coordinates (0, 0). The scan area coordinates are referenced, in dots, from that position, using the specified horizontal and vertical resolutions, determined by the values of the `xDpi` and `yDpi` fields. Figure 4-1 on page 70 shows a sample document scan area with the coordinates (10, 4) and (26, 16).

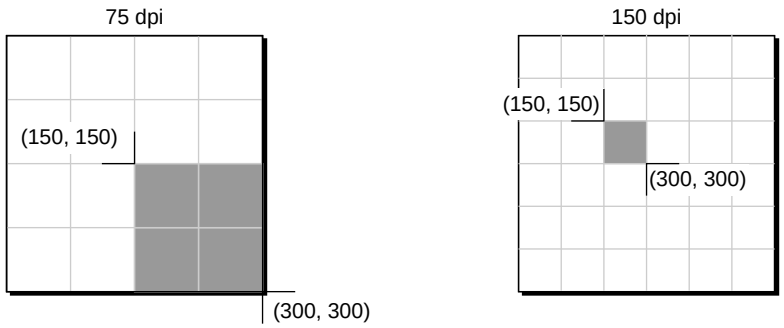
Figure 4-1 A scan rectangle



In Figure 4-1 the dark gray area indicates the desired image data and the light gray area indicates the extra data the scanner also returns. Even though pixels to the left and right of the desired scan area are not really part of the scan area, all the pixels up to the byte boundary (shown by the byte width value in the figure) are returned. Because the scanner always returns data in 1-byte quantities, your application program must mask off these excess pixels if you wish to read scan areas that are not byte-aligned directly into existing bitmaps. Calculate the byte width and height from the scan area coordinates, so that your application program knows how much data will be returned and how wide each line will be.

The area that the scanner actually scans depends on both the scan area and the resolution. Figure 4-2 illustrates how the scan area and the resolution interact to define the size and location of the area that is actually scanned. (In the figure, each square represents 1 square inch of the scan area.) In both diagrams, the scan area has been set to (150, 150), (300, 300), which is defined by the parameter `scanRect`. The only difference between the two diagrams is the resolution. Note that although the position and size of the area being scanned have changed, the amount of data returned for each area remains the same.

Figure 4-2 Scan areas at different resolutions



`brightness` This field controls the brightness of the resulting image. The scanner maps the visible spectrum over numeric values from 1 to either 255, or a maximum value specified by the `brightnessMax` field in the appropriate `ScCompRec` record returned by the `ScGetStdFeatures` function. (See the description of `ScGetStdFeatures`, in Chapter 3, "Scanner Driver Functions.") The value of the `brightness` field selects a point in that range. Lower numeric values translate to darker picture elements. Higher values translate to lighter elements.

You determine the range of valid values for the `brightness` field by examining the `brightnessRange` field in the appropriate `ScCompRec` record. If the value of the `brightnessRange` field is 0, you must set the `brightness` field to a value ranging from 1 to the value of the `brightnessMax` field in the `ScCompRec` record. If the value of the `brightnessRange` field is not 0, you may set the `brightness` field to a value ranging from 1 to 255, or from 1 to the value of the `brightnessMax` field. If you use values greater than the value of `brightnessMax`, you must set the high-order bit of the `brightness` field to 1.

In summary, if the value of the `brightnessRange` field is 0, then valid values for the `brightness` field range from 0 to the value of `brightnessMax`. If `brightnessRange` is not set to 0, then values of `brightness` may range from 0 to the value of `brightnessRange` or from $8000 to $80FF.

The scanner driver interprets this value in different ways, depending upon the specified composition mode. In Line Art mode, `brightness` determines the threshold level at which a pixel goes from black to white. The scanner renders as black pixels those gray values that are higher than the brightness threshold. It renders those that are lower than the threshold as white pixels. Increasing the value of this field increases the overall brightness level of the resulting image (because it raises the threshold at which an element goes from light to dark). Decreasing the value of this field decreases the overall brightness level. A value of 0 selects a default brightness value in the scanner.

In Grayscale and Halftone modes, the `brightness` field positions the center point of the visible spectrum within the supported range of values. The scanner driver then divides portions of the two resulting parts into a number of gray levels. Your application uses the `contrast` field to control the amount of the visible spectrum devoted to gray levels. Higher `brightness` values yield a lighter image, because more of the dynamic range is devoted to lighter shades of gray. Lower values yield a darker image.

`contrast` This field interacts with the `brightness` field to control the contrast of the scanned image rendered in Halftone or Grayscale mode. The scanner divides a portion of the visible spectrum into a number of gray-level segments of equal size. The `contrast` field determines the amount of the visible spectrum devoted to gray levels on either side of the spectrum's center point (which is specified by the `brightness` field). Those portions of the spectrum that lie outside the gray areas are rendered as black or white.

You determine the range of valid values for the `contrast` field by examining the `contrastRange` field in the appropriate `ScCompRec` record. If the value of the `contrastRange` field is 0, you must set the `contrast` field to a value ranging from 1 to the value of the `contrastMax` field in the `ScCompRec` record returned by the `ScGetStdFeatures` function. (See the description of `ScGetStdFeatures`, in Chapter 3.) If the value of the `contrastRange` field is not 0, you may set the `contrast` field to a value ranging from 1 to 255, or from 1 to the value of the `contrastMax` field. If you use values greater than the value of `contrastMax`, you must set the high-order bit of the `contrast` field to 1.

In summary, if the value of the `contrastRange` field is 0, then valid values for the `contrast` field range from 0 to the value of `contrastMax`. If `contrastRange` is not set to 0, then `the` values of `contrast` may range from 0 to the value of `contrastRange` or from $8000 to $80FF.

In the Color OneScanner, increasing the value of the contrast parameter decreases the contrast level, because the scanner renders a greater portion of the visible spectrum as gray tones. Decreasing the value of this parameter increases the contrast level. A value of 0 selects a default contrast value in the scanner. This parameter is valid only in Halftone and Grayscale modes.

In the Apple Scanner and the OneScanner, increasing the parameter increases the contrast level, and decreasing the parameter decreases the contrast level.

`composition`

This field specifies the composition mode for the region to be scanned and determines the type of data to be acquired. The composition modes are as follows:

Composition mode	Composition value
Line Art mode	`scLineArt`
Halftone mode	`scHalfTone`
Grayscale mode	`scGrayScale`
Bi-level Color mode	`scBilevelColor`
Full Color mode	`scFullColor`

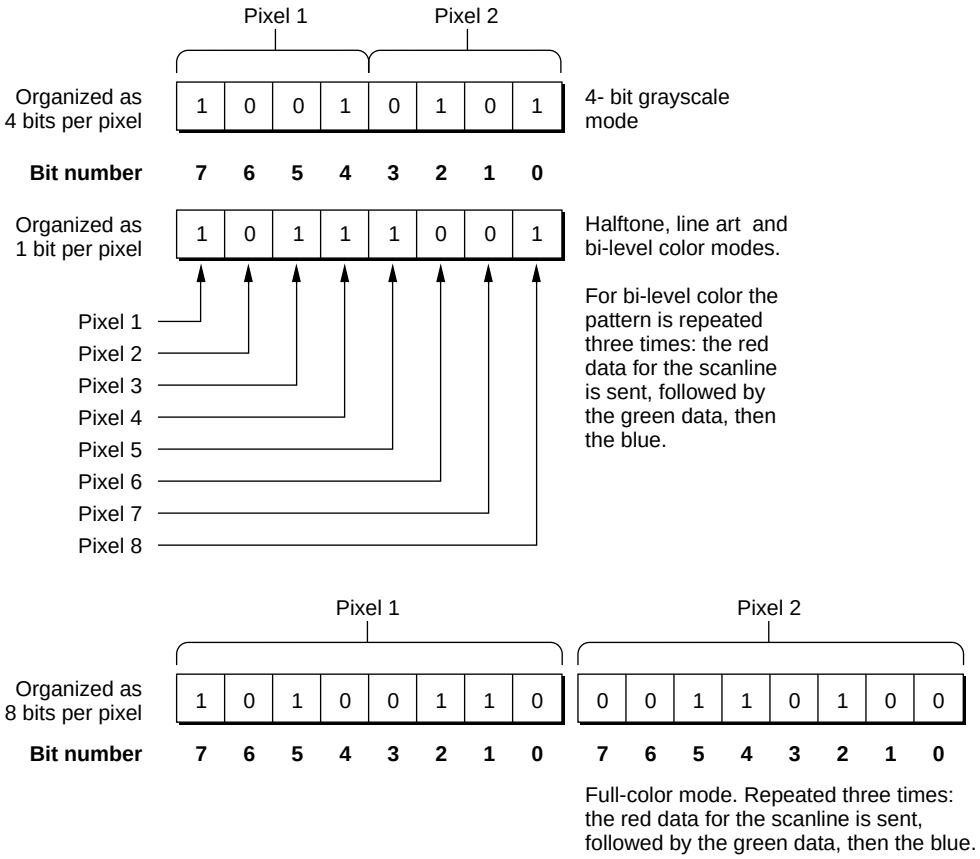
`bitsPerPixel`

This field determines the number of bits required to represent one pixel of data returned from the scanner. Valid values are determined by the setting of the `bitsPerPixel` field in the appropriate `ScCompRec` record returned by the `ScGetStdFeatures` function. (See the description of `ScGetStdFeatures`, in Chapter 3.)

The graphics data bits are packed in the return byte to the closest power of 2, and they are always left-aligned with the remaining bits set to 0, as shown in Figure 4-3 on page 74. For example, when you set the `bitsPerPixel` field to 4 (for 4-bit Grayscale mode), two pixels share one byte, when you set the `bitsPerPixel` field to 1 (Halftone and Line Art modes), eight pixels share one byte. When you set the parameter bits per pixel to 24, one pixel occupies three bytes.

`halfTone` This field specifies the halftone pattern that the scanner uses to render halftone images. This field contains an index into the array of halftone pattern names in the `ScHalfToneArray` structure returned by the `ScGetHalfTones` function, or a special value. (See the description of `ScGetHalfTones`, in Chapter 3.) To select a specific pattern, set this field to the 1-relative index of the entry containing the name of the desired pattern. A value of 0 selects the default halftone pattern (a 4-by-4 spiral). A value of –1 selects a pattern your application previously downloaded to the scanner driver. (See the description of `ScSetHTPattern`, in Chapter 3.)

Figure 4-3 Orientation of returned data bits



ScCompRec

The ScCompRec record contains information describing a composition mode. Composition modes (or scan modes) indicate the mode used by the scanner. Apple scanners support such composition modes as Line Art, Halftone, and Grayscale. (See "Composition," in Chapter 1, for more information.)

The ScGetStdFeatures driver function returns information about compositions supported by the attached scanner in an array of ScCompRec records formatted within the returned ScStdFeaturesRec record. By examining those ScCompRec records, your application can determine the valid composition modes for the attached scanner and the valid parameter ranges for the supported types.

The following Pascal code shows the layout of the ScCompRec record:

```
TYPE  ScCompRec = PACKED RECORD
         brightnessRange : Byte;
         contrastRange : Byte;
         reserved : Byte;
         resFlags : Byte;
         resElements : Integer;
         halfToneElements : Integer;
         brightnessMax : Integer;
         contrastMax : Integer;
         bitsPerPixel : LongInt;
         minReadSize : Integer;
      END;
```

The fields are defined as follows:

`brightnessRange`
: This field indicates the number of brightness levels available on the attached scanner. If the value of the `brightnessRange` field is 0, you must set the `brightness` field in any `ScAreaRec` records passed to the `ScSetScanArea` driver function to a value ranging from 1 to the value of the `brightnessMax` field. If the value of the `brightnessRange` field is not 0, you may set the `brightness` field to a value ranging from 1 to 255 or from 1 to the value of the `brightnessMax` field. If you use values greater than the value of `brightnessMax`, you must set the high-order bit of the `brightness` field to 1.

`contrastRange`
: This field indicates the number of contrast levels available on the attached scanner. If the value of the `contrastRange` field is 0, you must set the `contrast` field in any `ScAreaRec` records passed to the `ScSetScanArea` driver function to a value from 1 to the value of the `contrastMax` field. If the value of the `contrastRange` field is not 0,

you may set the `contrast` field to a value from 1 to 255, or from 1 to the value of the `contrastMax` field. If you use values greater than the value of `contrastMax`, you must set the high-order bit of the `contrast` field to 1.

`reserved` This field is unused. It is set to 0.

`resFlags` This field contains bit flags that define scanner capabilities:

1 dpi	bit 0	Indicates whether the scanner supports resolution in increments of 1dpi. The value of this flag affects the way your application should interpret the `ScResArray` structure returned by the `ScGetRes` driver function. (See the description of `ScGetRes`, in Chapter 3.) 0 = Scanner does not support increments of 1 dpi 1 = Scanner supports increments of 1 dpi

`resElements`

This field indicates the total number of elements in the `ScResArray` structure returned by the `ScGetRes` driver function for this composition mode. (See the description of `ScGetRes`, in Chapter 3.) The elements in the `ScResArray` structure specify the resolutions supported by the attached scanner in this composition mode.

The driver sets this field to 0 for unsupported composition modes. Use this value as a flag to indicate that a particular mode is not supported by the attached scanner.

`halfToneElements`

This field indicates the total number of elements in the `ScHalfToneArray` structure returned by the `ScGetHalfTones` driver function for the selected mode. (See the description of `ScGetHalfTones`, in Chapter 3.) The elements in the `ScHalfToneArray` structure specify the names of the halftone patterns supported by the attached scanner.

The driver sets this field to 0 for composition modes (such as Line Art and Grayscale) that do not support halftone patterns.

`brightnessMax`

This field indicates the number of brightness levels available on the attached scanner in cases where the `brightnessRange` field is set to 0. Your application should then use this value as the upper limit for the value of the `brightness` field in any `ScAreaRec` records passed to the `ScSetScanArea` driver function. If `brightnessRange` is not set to 0, then your application may set the `brightness` field to values up to 255. In this case, your application must also set the high-order bit of the `brightness` field to 1.

`contrastMax`

This field indicates the number of contrast levels available on the attached scanner in cases where the `contrastRange` field is set to 0. Your application should then use this value as the upper limit for the value of the `contrast` field in any `ScAreaRec` records passed to the `ScSetScanArea` driver function. If `contrastRange` is not set to 0, then your application may set the `contrast` field to values up to 255. In this case, your application must also set the high-order bit of the `contrast` field to 1.

`bitsPerPixel`

This field contains bit flags indicating the number of bits that may be used to encode a single pixel in the given composition mode. Note that more than one flag may be set to 1, indicating that the attached scanner supports more than one encoding option for that composition mode. In such cases, your application must use the `bitsPerPixel` field of the `ScAreaRec` record passed to the `ScSetScanArea` driver function to indicate the pixel encoding option for the scan. (See the description of `ScSetScanArea`, in Chapter 3.)

The bits are defined as follows:

bit 0	1 bit per pixel available
bit 1	2 bits per pixel available
bit 2	3 bits per pixel available
bit 3	4 bits per pixel available
.	
.	
.	
bit 30	31 bits per pixel available
bit 31	32 bits per pixel available

A bit is set to 1 if the corresponding encoding option is supported.

`minReadSize`

This field indicates the minimum number of data bytes your application should read with each call to the `ScDoScan` function. (See the description of `ScDoScan`, in Chapter 3.) If your application issues read requests for less data than is specified by this parameter, or if your application calls `ScDoScan` too infrequently, the scanner may not operate smoothly for every scan.

ScHalfToneArray

The `ScHalfToneArray` structure contains a list of halftone pattern names. The `ScGetHalfTones` driver function returns the list of supported halftone patterns in an `ScHalfToneArray` structure. (See the description of `ScGetHalfTones`, in Chapter 3, "Scanner Driver Functions.") Each entry in the array corresponds to a given halftone pattern. The pattern name is stored in a Pascal string that can be no longer than 31 characters.

When your application requests a scan in Halftone mode, it must specify the halftone pattern for the scan. You identify the halftone pattern name by supplying its index in the returned array of supported patterns in the `halfTone` field of the `ScAreaRec` record. (See the descriptions of `ScAreaRec` and `ScSetScanArea`, in Chapter 3.)

The following Pascal code shows the format of the `ScHalfToneArray` structure.

```
TYPE  String31 = String[31];
      ScHalfTonePtr = ^ ScHalfToneArray;
      ScHalfToneArray = ARRAY[1..1] OF String31;
```

The data types are defined as follows:

`String31` This data type is a 31-character Pascal string containing the name of a supported halftone pattern.

`ScHalfTonePtr`
This data type defines a pointer to a halftone array stored as an `ScHalfToneArray` structure.

`ScHalfToneArray`
This data type defines the halftone array itself. An `ScHalfToneArray` structure contains entries defining halftone patterns. Each entry in the array corresponds to a halftone pattern and contains the pattern name stored as a Pascal-formatted string of up to 31 characters in length. Table 4-1 shows a sample of halftone array elements.

Table 4-1 Samples of halftone array elements

Halftone array element index	Element contents
1	"4-by-4 matrix, Spiral"
2	"4-by-4 matrix, Bayer"
3	"5-by-5 matrix, Spiral"
.	.
.	.
.	.
n	Halftone matrix *n*

ScResArray

The `ScResArray` record contains a list of scanning resolutions. Each resolution is represented by an integer corresponding to the dots per inch (dpi) measure of the resolution. For example, 300 dpi would be represented in an `ScResArray` entry by a value of 300. The `ScGetRes` driver function returns the list of resolutions supported by the attached scanner in an `ScResArray` record. (See the description of `ScGetRes`, in Chapter 3.)

The following Pascal code shows the format of the `ScResArray` record:

```
TYPE  ScResPtr = ^ScResArray;
      ScResArray = ARRAY[1..1] OF Integer;
```

The data types are defined as follows:

`ScResPtr` This data type defines a pointer to a resolution array stored as an `ScResArray` record.

`ScResArray` This data type defines the resolution array itself. An `ScResArray` record contains entries defining resolutions. Each entry in the array corresponds to a given resolution, and the entry contains the resolution value expressed as an integer corresponding to the dpi measure for the resolution. In addition, for scanners that support resolution in increments of 1 dpi (see the preceding description of the `resFlags` field of the `ScCompRec` record), the first and last values in the array specify the lower and upper limits of resolution, respectively. Note that, in this case, additional array elements may be included for convenience in building resolution lists that can be displayed.

ScScanAreaRec

The ScScanAreaRec record defines all the areas to be processed in a given scan. Your application formats the ScScanAreaRec record and passes a pointer to that formatted record to the ScSetScanArea driver function. (See the description of ScSetScanArea, in Chapter 3.) The array of ScAreaRec records contained in the ScScanAreaRec record defines the individual scan areas.

The following Pascal code shows the layout of the ScScanAreaRec record:

```
TYPE  ScScanAreaPtr = ^ScScanAreaRec;
      ScScanAreaRec = RECORD
         reserved : LongInt;
         numAreas : Integer;
         scanAreas : ARRAY[1..1] OF ScAreaRec;
      END;
```

The fields and data types are defined as follows:

ScScanAreaPtr
: This data type defines a pointer to a scan area record stored as an ScScanAreaRec record.

reserved
: This field is reserved for future use. Set it to 0.

numAreas
: This field indicates the number of individual scan areas defined for the scan, and it therefore corresponds to the number of ScAreaRec records specified by the ScanAreas field. Your application must set numAreas to a value ranging from 1 to the maximum number of secondary areas supported by the attached scanner plus 1. The maximum number of secondary areas is given in the secondaryMax field of the ScAdvFeaturesRec record returned by the ScGetAdvFeatures function. (See the description of ScGetAdvFeatures, in Chapter 3.)

scanAreas
: This array of ScAreaRec records defines the individual scan areas for the scan operation. Each area to be scanned must be defined with its own ScAreaRec record. The numAreas field indicates the number of ScAreaRec records in the array. The first ScAreaRec record in the array defines the primary scan area. Any subsequent array entries define secondary scan areas. Secondary scan areas are supported on the Apple Scanner. (See the description of ScGetAdvFeatures, in Chapter 3.)

ScStdFeaturesRec

The `ScStdFeaturesRec` record contains information describing the capabilities of the attached scanner. The `ScGetStdFeatures` driver function returns information about the capabilities of the attached scanner in an `ScStdFeaturesRec` record. (See the description of `ScGetStdFeatures`, in Chapter 3.)

The following Pascal code shows the format of the `ScStdFeaturesRec` record:

```
TYPE  ScStdFeaturesPtr = ^ ScStdFeaturesRec;
      ScStdFeaturesRec = RECORD
         scannerType : ResType;
         version : Integer;
         scanWidthNum : Integer;
         scanWidthDen : Integer;
         scanLengthNum : Integer;
         scanLengthDen : Integer;
         composition : ARRAY[scLineArt..scFullColor) OF ScCompRec;
      END;
```

The fields and data type are defined as follows:

`ScStdFeaturesPtr`
: This data type defines a pointer to an `ScStdFeaturesRec` record.

`scannerType`
: This field identifies the type of the attached scanner:

APL4	Apple Scanner
APL8	OneScanner
APLC	Color OneScanner

`version` This field indicates the structure version number.

`scanWidthNum`
: This field contains the numerator of the scan area width, which is described later in this section.

`scanWidthDen`
: This field contains the denominator of the scan area width, which is described later in this section.

`scanLengthNum`
: This field contains the numerator of the scan area length, which is described later in this section.

`scanLengthDen`
: This field contains the denominator of the scan area length, which is described later in this chapter.

The `scanWidthNum`, `scanWidthDen`, `scanLengthNum`, and `scanLengthDen` fields specify the maximum size of the scan area by defining its width and length in inches. The width numerator, `scanWidthNum`, divided by the width denominator, `scanWidthDen`, gives the width in inches. The length numerator, `scanLengthNum`, divided by the length denominator, `scanLengthDen`, gives the length in inches. Your application must never define a scan area that extends beyond these boundaries.

For example, the field values for a scanner that can accommodate an 8.5-by-14-inch page are as follows:

Field	Value
`scanWidthNum`	17
`scanWidthDen`	2
`scanLengthNum`	14
`scanLengthDen`	1

Therefore, at 75 dpi, the maximum width setting allowable in the `scanRect` field of any `ScAreaRec` records passed to the `ScSetScanArea` driver function is given by the following formula:

([(75 ∗ 17) / 2] / 2) ∗ 2 = 636

Note that the value ([75 ∗ 17] / 2) was truncated to an even byte boundary. Be sure to discard the remainder from all division operations.

`composition`

This field contains an array of `ScCompRec` records, each of which defines the characteristics of the support offered for a single composition mode. Your application can access the `ScCompRec` record for a particular composition mode by using the appropriate constant for that mode as an index into this array. Here are the possible composition modes and their corresponding index values:

Composition mode	Index value
Line Art mode	`scLineArt`
Halftone mode	`scHalfTone`
Grayscale mode	`scGrayScale`
Bi-level Color mode	`scBilevelColor`
Full Color mode	`scFullColor`

If a scanner does not support a particular composition mode, the driver sets the `resElements` field in the appropriate `ScCompRec` record to 0.

Advanced Data Structures

This section describes the data structures your application uses to interact with advanced scanner driver functions. (See "Advanced Functions,", in Chapter 3, for descriptions of these driver functions).

The structures described in this section are:

- `ScAdvFeaturesRec`
- `ScPatRec`
- `ScGammaTableRec`
- `ScMatrix`

ScAdvFeaturesRec

The `ScAdvFeaturesRec` record contains information describing the advanced capabilities of the attached scanner. The `ScGetAdvFeatures` driver function returns information about the advanced features of the attached scanner in an `ScAdvFeaturesRec` record. (See the description `ScGetAdvFeatures`, in Chapter 3.)

The following Pascal code shows the format of the `ScAdvFeaturesRec` record:

```
TYPE  ScAdvFeaturesPtr = ^ScAdvFeaturesRec;
      ScAdvFeaturesRec = RECORD
         reserved : LongInt;
         version : Integer;
         secondaryMax : Integer;
         downLoadFlags : LongInt;
         restrictFlags : LongInt;
         controlFlags : LongInt;
      END;
```

The fields and data type are defined as follows:

`ScAdvFeaturesPtr`
: This data type defines a pointer to an `ScAdvFeaturesRec` record.

`reserved`
: This field is reserved for future expansion.

`version`
: This field indicates the structure version number.

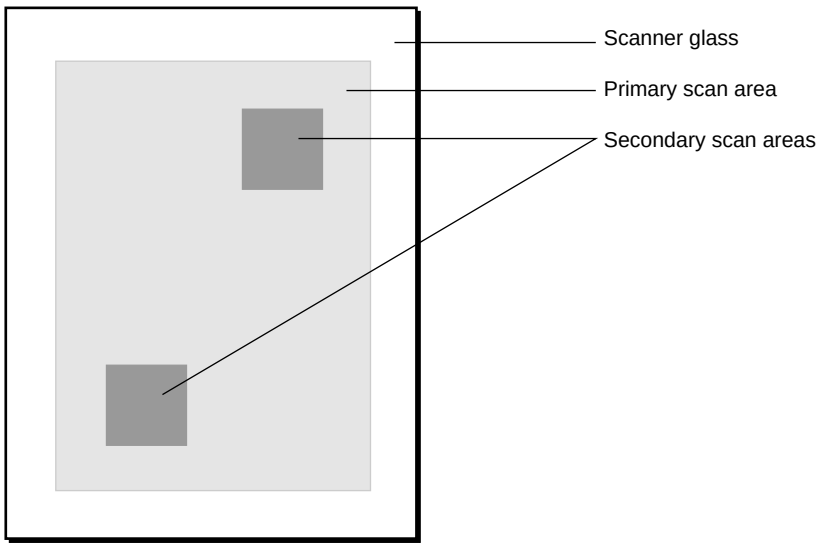
`secondaryMax`
: This field indicates whether the scanner supports secondary scan areas and, if so, how many of them. If the scanner does not support secondary areas, the driver sets the field to 0. If the scanner does support secondary areas, the parameter contains the maximum number of secondary areas allowed.

  Secondary scan areas are "cut out" of the primary scan area and can accommodate some differences in scan parameters. For example, your application can use this feature to mix line art data and halftone data in one document.

  Your application must comply with several restrictions to use secondary scan areas successfully. The secondary scan areas must lie within the primary scan area and must have the same horizontal and vertical resolution settings as the primary area. In addition, all secondary areas must use the same composition mode, contrast, brightness, and halftone parameters. Figure 4-4 shows the relationship between a primary scan area and secondary scan areas.

  Your application program specifies secondary scan areas when defining the scan area with the `ScSetScanArea` function. (See the description of `ScSetScanArea`, in Chapter 3.)

Figure 4-4 Primary and secondary scan areas



`downLoadFlags`

This field indicates the halftone matrix dimensions supported by the attached scanner. These matrix dimensions apply to the halftone patterns your application downloads to the scanner with the `ScSetHTPattern` driver function. (See the description of `ScSetHTPattern`, in Chapter 3.) If `downLoadFlags` is nonzero, then the attached scanner supports the `ScSetHTPattern` function. Each matrix dimension corresponds to a bit in `downLoadFlags`, as shown here. For supported dimensions, the driver sets the corresponding bit to 1.

bit 0	2-by-2 matrix
bit 1	3-by-3 matrix
bit 2	4-by-4 matrix
.	
.	
.	
bit 14	16-by-16 matrix
bits 15–31	Reserved

Some scanners may support asymmetrical halftone matrixes, where the x and y matrix dimensions differ. For scanners that support asymmetrical matrixes, the driver sets bit 3 of the `restrictFlags` field to 1. In addition, the driver sets the bits in the `downLoadFlags` field to indicate the valid range of values for both the x and y matrix dimensions. For example, if bit 3 of `restrictFlags` is set to 1, and bits 0, 1, and 2 in `downloadFlags` are set to 1, then the attached scanner supports 2-by-2, 3-by-3, 4-by-4, 2-by-3, 2-by-4, 3-by-2, 3-by-4, 4-by-2, and 4-by-3 matrixes.

`restrictFlags`

This field indicates restrictions that your application program may safely ignore when setting the scanner control parameters with the `ScSetScanArea` function. (See the description of `ScSetScanArea`, in Chapter 3.) Each restriction type corresponds to a bit in `restrictFlags`, as shown here. If the driver sets a bit to 1, your application may take advantage of the corresponding feature.

bit 0	Horizontal and vertical resolution settings may be of different values
bit 1	Secondary scan areas may be of the same type as the primary scan area
bit 2	Automatic background adjustment is available in Line Art mode. Set the `brightness` field in the appropriate `ScAreaRec` record to –1 to enable automatic background adjustment
bit 3	Downloaded halftone matrixes may have different x and y dimensions. (See the description of `ScSetHTPattern`, in Chapter 3.)
bits 4–31	Reserved for future use. Do not use

`controlFlags`

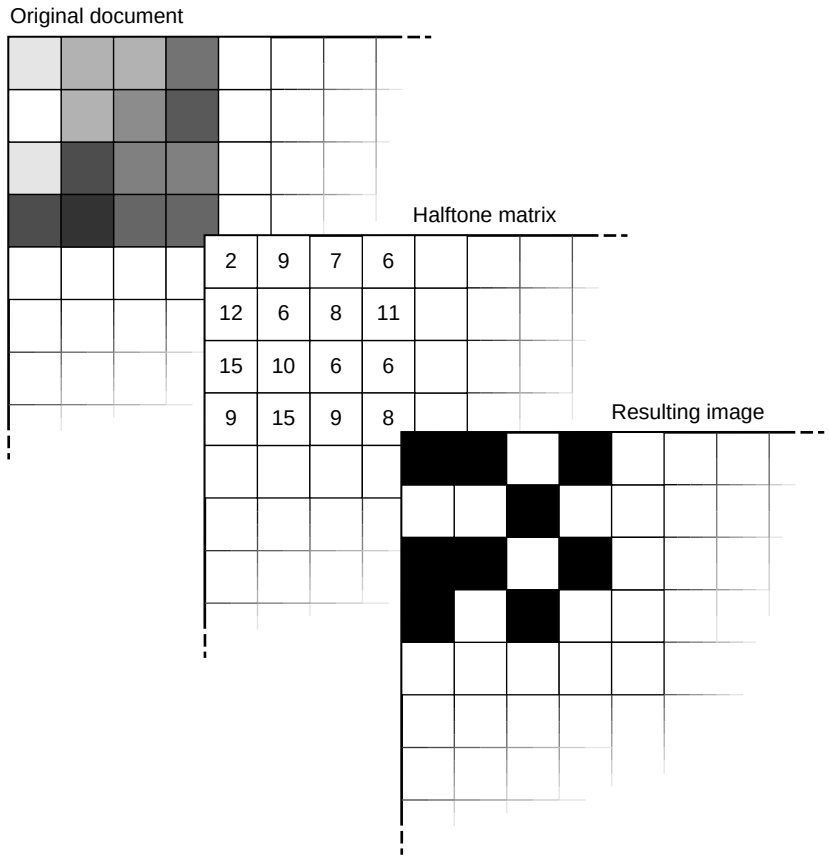
This field indicates the advanced driver functions that the attached scanner supports. Each bit in `controlFlags` corresponds to an advanced function, as shown here. If the attached scanner supports an advanced driver function, the driver sets the corresponding bit to 1.

bit 0	`ScSetGroup3` function supported
bit 1	`ScSetGraymap` function supported
bit 2	`ScSetThreshold` function supported
bit 3	`ScSetLamp` function supported
bit 4	`ScSetNoHome` function supported
bit 5	`ScSetWaitButton` function supported
bit 6	`ScSetSpeed` function supported
bit 7	`ScSetLed` function supported
bit 8	`ScGetButton`, `ScResetButton` functions supported
bit 9	`ScSetNoCal` function supported
bit 10	`ScLoadGamma` function supported
bit 11	`ScLoadMatrix` function supported
bit 12	`ScInvertPixels` function supported
bit 13	`ScScannerSetAtoD` function supported
bit 14	`ScSensorSelect` function supported
bits 15-31	Reserved for additional control call functions. These bits are set to 0.

ScPatRec

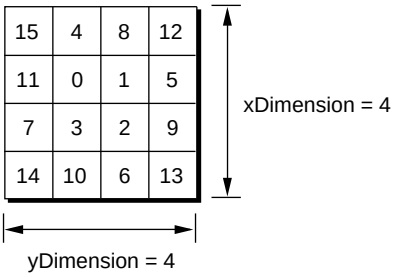
The `ScPatRec` record defines a pattern for generating scan images in Halftone mode. The halftone pattern consists of a matrix of brightness values. In Halftone mode, the scanner filters scanned data through that matrix, turning resulting pixels on or off according to the brightness of the scanned pixel relative to the appropriate matrix value. Figure 4-5 shows this process. The resulting two-level data emulates gray-scale encoding in much the same way that dithering a color image fools the eye into seeing colors that are not there.

Figure 4-5 Halftone processing



Your application loads a custom halftone pattern matrix into the attached scanner by formatting an `ScPatRec` record and passing that record to the `ScSetHTPattern` driver function. (See the description of `ScSetHTPattern`, in Chapter 3.) Figure 4-6 on page 88 shows the brightness threshold values for a 4-by-4 spiral matrix.

Figure 4-6 The electronic halftone matrix for a 4-by-4 spiral pattern



The following Pascal code shows the format of the `ScPatRec` record:

```
TYPE  ScPatPtr = ^ScPatRec;
      ScPatRec = PACKED RECORD
         xDimension : Byte;
         yDimension : Byte;
         patData : PACKED ARRAY[1..64] OF Byte;
      END;
```

The fields and data type are defined as follows:

`ScPatPtr` This data type defines a pointer to an `ScPatRec` record.

`xDimension` This field specifies the horizontal (x-axis) dimension of the halftone matrix. The value of this field must meet the restrictions defined by the `downLoadFlags` field in the `ScAdvFeaturesRec` record returned by the `ScGetAdvFeatures` function.

`yDimension` This field specifies the vertical (y-axis) dimension of the halftone matrix. The value of this field must meet the restrictions defined by the `downLoadFlags` field in the `ScAdvFeaturesRec` record returned by the `ScGetAdvFeatures` function.

`patData` This one-dimensional array contains the halftone pattern matrix. Your application concatenates rows of values from the matrix into this array. The driver then rebuilds the original matrix from these values, using the dimensions specified by the `xDimension` and `yDimension` fields. The values in `patData` specify relative brightness thresholds in the range of 0 to (`xDimension * yDimension`) – 1 in the Apple Scanner, and a range of 0 to 255 in the OneScanner. Figure 4-6 presents a 4-by-4 matrix with the 16 threshold values set to generate a spiral pattern.

ScGammaTableRec

`ScGammaTableRec` loads a table of values used to change the scanner's intensity curve. Each scanned pixel value is used as an address for the table, and the three bytes from the table (R, G, and B) at that address are returned as the pixel's intensity.

The following Pascal code shows the format of the `ScLoadGamma` record:

```
TYPE  ScGammaTablePtr = ^ScGammaTableRec;
      ScGammaTableRec = PACKED RECORD
         RedGamma : ARRAY[1..256] OF BYTE;
         GreenGamma : ARRAY[1..256]OF BYTE;
         BlueGamma : ARRAY[1..256]OF BYTE;
      END
```

The fields and data type are defined as follows:

`ScGammaTablePtr`
: This data type defines a pointer to an `ScGammaTableRec` record.

`RedGamma`
: This field contains a table of intensity values for the Red channel.

`GreenGamma`
: This field contains a table of intensity values for the Green channel.

`BlueGamma`
: This field contains a table of intensity values for the Blue channel.

ScMatrix

`ScMatrix` loads the matrix multiplier with the contents of the array data. The Color OneScanner multiplier is 16 bits: 15 bits of data, which should be left justified, and one sign bit.

The following Pascal code shows the format of the `ScMatrix` record:

```
TYPE  ScMatrixPtr = ^ScMatrix;
      ScMatrix : ARRAY[1..9]OF INTEGER;
```

The fields and data types are defined as follows:

`ScMatrixPtr`
: This data type defines a pointer to an `ScMatrix` record.

`ScMatrix`
: The `ScMatrix` field contains signed integer values to be loaded into the matrix multiplier. Data should be left-justified.

For further information, see Appendix B, “Optimizing the Color OneScanner.”

Scanner Driver Summary

Scanner Driver Summary

This chapter summarizes the constant values, data types, and functions for the standard and advanced features available in the scanner driver.

Standard Constants

The following constants define values used in standard driver functions.

```
CONST
   scLineArt      = 0;
   scHalfTone     = 1;
   scGrayScale    = 2;
   scBilevelColor = 3;
   scFullColor    = 4;
```

Advanced Constants

The following constants define values used in advanced driver functions.

```
CONST
   scLightDetail  = 0;  { GrayMap 0 }
   scNormalDetail = 1;  { GrayMap 1 }
   scDarkDetail   = 2;  { GrayMap 2 }

   scSpeedNormal  = 0;  { NORMAL Speed, slow with
                          handshaking }
   scSpeedHigh = 1;     { HIGH Speed, fast with
                          handshaking }
   scSpeedFast = 2;     { FAST Speed, fast without
                          handshaking }

{ Apple (Vendor) Unique  Param Type Constants }

   scUniqueUnpark    = 0;     {Unpark the carriage }
   scUniquePark      = 1;     {Park the carriage }
   scUniqueAbsPos    = 2;     {Absolute carriage positioning }
   scUniqueRelPos    = 3;     {Relative carriage positioning}
   scUniqueSetCRAM   = 5;     {Set CRAM Data (always 2550 bytes)}
   scUniqueGetCRAM   = 6;     {Get CRAM Data (always 2550 bytes)}
```

```
   scAllSensor       = 0x00
   scRedSensor       = 0x01
   scGreenSensor     = 0x02
   scBlueSensor      = 0x03

Control Flags

scSetGroup3          = 1;
scSetGamma           = 2;
scSetThreshold       = 4;
scSetLampOn          = 8;
scSetNoHome          = 16;
scSetWait            = 32;
scSetSpeed           = 64;
scSetLed             = 128;
scGetButton          = 256;
scSetNoCal           = 512;
scLoadGamma          = 1024;
scLoadMatrix         = 2048;
scInvertPixels       = 4096;
scSetAtoD            = 8192
scSensorSelect       = 16384
```

Standard Data Types

The following data types are used by standard driver functions:

```
ScAreaRec = RECORD
      reserved : LongInt;
      xDpi : Integer;
      yDpi : Integer;
      scanRect : Rect;
      brightness : Integer;
      contrast : Integer;
      composition : SignedByte;
      bitsPerPixel : SignedByte;
      halfTone : Integer;
   END;
```

Scanner Driver Summary

```
TYPE
ScCompRec = PACKED RECORD
   brightnessRange : Byte;
   contrastRange : Byte;
   reserved : Byte;
   resFlags : Byte;
   resElements : Integer;
   halfToneElements : Integer;
   brightnessMax : Integer;
   contrastMax : Integer;
   bitsPerPixel : LongInt;
   minReadSize : Integer;
END;

String31 = STRING[31];
ScHalfTonePtr = ^ScHalfToneArray;
ScHalfToneArray = ARRAY[1..1] OF String31;
ScResPtr = ^scResArray;
ScResArray = ARRAY[1..1] OF Integer;
ScScanAreaPtr = ^ScScanAreaRec;
ScScanAreaRec = RECORD
   reserved : LongInt;
   numAreas : Integer;
   scanAreas : ARRAY[1..1] OF ScAreaRec;
END;

ScStdFeaturesPtr = ^ScStdFeaturesRec;
ScStdFeaturesRec = RECORD
   scannerType : ResType;
   version : Integer;
   scanWidthNum : Integer;
   scanWidthDen : Integer;
   scanLengthNum : Integer;
   scanLengthDen : Integer;
   composition : ARRAY[scLineArt..scFullColor+1] OF
      ScCompRec;
END;
```

Advanced Data Types

The following data types are used by advanced driver functions. For further information see the section "`ScAdvFeaturesRec`," in Chapter 4.

```
TYPE
ScAdvFeaturesPtr = ^ScAdvFeaturesRec;
ScAdvFeaturesRec = RECORD
   reserved : LongInt;
   version : Integer;
   secondaryMax : Integer;
   downLoadFlags : LongInt;
   restrictFlags : LongInt;
   controlFlags : LongInt;
END;

ScPatPtr = ^ScPatRec;
ScPatRec = PACKED RECORD
   xDimension : Byte;
   yDimension : Byte;
patData : PACKED ARRAY[1..64] OF Byte;
END;

ScGammaTablePtr = ^ scGammaTableRec;
   ScGammaTableRec = PACKED RECORD
            RedGamma : ARRAY[1..256}OF BYTE;
            GreenGamma : ARRAY[1..256] OF BYTE;
            BlueGamma : ARRAY[1..256] OF BYTE;
   END;

scMatrixPtr = ^ scMatrix
scMatrix : ARRAY[1..9] OF INTEGER
END;
```

Standard Functions

The following Pascal declarations define the standard driver functions:

```
FUNCTION ScAbortScan (refNum: Integer) : OSErr;
FUNCTION ScClose (refNum: Integer) : OSErr;
FUNCTION ScDoScan (refNum: Integer;
      buffer: Ptr;
      VAR count: LongInt;
      unused: Integer;
      byteWidth,rowBytes: Integer) : OSErr;
FUNCTION ScGetHalfTones (refNum: Integer;
      compType: Integer;
      halfTonePtr: ScHalfTonePtr) : OSErr;
FUNCTION ScGetRes (refNum: Integer;
      compType: Integer;
      resPtr: ScResPtr) : OSErr;
FUNCTION ScGetStdFeatures (refNum: Integer;
      stdFeaturesPtr: ScStdFeaturesPtr;
      length: Integer) : OSErr;
FUNCTION ScOpen (VAR refNum: Integer) : OSErr;
FUNCTION ScSetScanArea (refNum: Integer;
      scanAreaPtr: ScScanAreaPtr) : OSErr;
```

Advanced Functions

The following Pascal declarations define the advanced driver functions:

```
FUNCTION ScGetAdvFeatures (refNum: Integer;
      advFeaturesPtr: ScAdvFeaturesPtr;
      length: Integer) : OSErr;
FUNCTION ScGetButton (refNum: Integer;
      VAR button: Boolean) : OSErr;
FUNCTION ScInvertPixels (refNum: Integer;
InvertFlag:BOOLEAN:OSErr;
FUNCTION ScLoadGamma (refNum: Integer,GammaTable :
scGammaTablePtr) :OSErr;
FUNCTION ScLoadMatrix (refNum: Integer;Matrix : scMatrixPtr) :
OSErr;
```

```
FUNCTION ScResetButton (refNum: Integer;
      setTrue: Boolean) : OSErr;
FUNCTION ScSensorSelect (refNum: Integer; sensor: INTEGER) :
OSErr;
FUNCTION ScSetGrayMap (refNum: Integer;
      grayMap: Integer) : OSErr;
FUNCTION ScSetGroup3 (refNum: Integer;
      compressOn: Boolean) : OSErr;
FUNCTION ScSetHTPattern (refNum: Integer;
      patPtr: ScPatPtr) : OSErr;
FUNCTION ScSetLamp (refNum: Integer;
      lampOn: Boolean) : OSErr;
FUNCTION ScSetLed (refNum: Integer;
      ledOn: Boolean) : OSErr;
FUNCTION ScSetNoCal (refNum: Integer;
      noCalMode: Boolean) : OSErr;
FUNCTION ScSetNoHome (refNum: Integer;
      noHome: Boolean) : OSErr;
FUNCTION ScSetScannerAtoD (refNum: Integer;Vrt,Vrb :BYTE) :OSErr;
FUNCTION ScSetSpeed (refNum: Integer;
      speed: Integer) : OSErr;
FUNCTION ScSetThreshold (refNum: Integer;
      thresholdLevel: Integer) : OSErr;
FUNCTION ScSetWaitButton (refNum: Integer;
      waitButton: Boolean) : OSErr;
FUNCTION ScVendorUnique (refNum: Integer;
      paramType: Integer; paramPtr: Ptr) : OSErr;
```

Function Result Codes

Table 5-1 lists the function result codes returned for the driver functions.

Note
There are two types of error code. The first type indicates driver errors and always has five digits. The second type indicates system errors and may have one through four digits. The two-digit codes shown in the following table indicate I/O system errors. ◆

Table 5-1 Result codes

Result	Code	Description
`noErr`	0	No error
`statusErr`	–18	Attached scanner does not support any advanced features
`badUnitErr`	-21	Scanner is not connected, or if connected, is not switched on
`openErr`	-23	Another program has opened the scanner driver
`scNotFoundErr`	–17064	Scanner was not found
		Scanner was not turned on
		Multiple SCSI devices have the same ID number
		Scanner has not completed the power-on initialization sequence
		SCSI cable was not properly connected
		SCSI bus was not properly terminated
		Scanner and scanner-driver revision were not correctly matched
`scComErr`	–17065	Communication-interface malfunction
		Scanner was turned off during scan
		SCSI cable was unplugged during scan
`scResetErr`	–17066	Scanner has been reset unexpectedly; parameters must be reset
`scParamErr`	–17067	Illegal parameter or command
		Driver received command for feature not supported by the scanner
`scScannerErr`	–17068	Internal scanner malfunction
		RAM or ROM failure
		Scanner needs servicing

continued

Table 5-1 Result codes (continued)

Result	Code	Description
`scLampErr`	–17069	Lamp is not on
		Lamp needs replacing
		CCD array malfunction
`scEOS`	–17070	End of scan (code occurs in response to `scDoScan` command when there is no data left to send)
`scDimLampErr`	–17071	The fluorescent light is functioning, but its output has dropped below 70 percent, and is too low for a scan to be accurate
		Lamp needs replacing
		The white target underneath the scanning bed is dirty (Color OneScanner only)
		The three mirrors are dirty (Color OneScanner only)
		The aperture plate is dirty (Color OneScanner only)
		The CCD is dirty or bad (Color OneScanner only)
		Scanning may continue if you receive result code –17071, but the quality of the output may not be satisfactory
`statusErr`	–18	Attached scanner does not support any advanced features
`scBusy`	–17072	Driver call made while scanner is busy (Color OneScanner only)

Device Manager Equivalents

Your application can invoke driver functions by calling the Macintosh Device Manager. Table 5-2 lists the Device Manager routines that are equivalent to the standard driver functions, and Table 5-3 lists the Device Manager routines for the advanced driver functions.

Table 5-2 Device Manager equivalents for standard driver functions

Driver function	Device Manager routine
`ScAbortScan`	`Control` routine with `csCode = 1 (killIO)`
`ScClose`	`Close` routine
`ScDoScan`	`Read` routine with `ioParam` set as follows: `ioBuffer = buffer`, `ioReqCount = count`, `ioActCount = count`, `ioPosMode = unused`, high word of `ioPosOffset = byteWidth`, and low word of `ioPosOffset = rowBytes`
`ScGetHalfTones`	`Status` routine with `csCode = 4`, `csParam = compType`, and `csParam + 2 = halfTonePtr`
`ScGetRes`	`Status` routine with `csCode = 3`, `csParam = compType`, and `csParam + 2 = resPtr`
`ScGetStdFeatures`	`Status` routine with `csCode = 2`, `csParam = stdFeaturesPtr`, and `csParam + 4 = length`
`ScOpen`	`Open` routine
`ScSetScanArea`	`Control` routine with `csCode = 2` and `csParam = scanAreaPtr`

Table 5-3 Device Manager equivalents for advanced driver functions

Driver function	Device Manager routine
`ScGetAdvFeatures`	`Status` routine with `csCode = 5`, `csParam = advFeaturesPtr`, and `csParam + 4 =` length
`ScGetButton`	`Control` routine with `csCode = 6` and `csParam =` pointer to `button`
`ScInvertPixels`	`Control` routine with `csCode = 15`, and `csParam = InvertFlag (Boolean)`
`ScLoadGamma`	`Control` routine with `csCode = 19`, and `csParam = scGammaTablePtr` (Ptr)
`ScLoadMatrix`	`Control` routine with `csCode = 18`, and `csParam = scMatrixPtr` (Ptr)
`ScResetButton`	`Control` routine with `csCode = 13` and `csParam = setTrue`
`ScSensorSelect`	`Control` routine with `csCode = 16`, and `csParam = sensor`
`ScSetGraymap`	`Control` routine with `csCode = 8` and `csParam = grayMap`
`ScSetGroup3`	`Control` routine with `csCode = 5` and `csParam = compressOn`
`ScSetHTPattern`	`Control` routine with `csCode = 4` and `csParam = patPtr`
`ScSetLamp`	`Control` routine with `csCode = 7` and `csParam = lampOn`
`ScSetNoCal`	`Control` routine with `csCode = 14` and `csParam = noCalMode`
`ScSetLed`	`Control` routine with `csCode = 12` and `csParam = ledOn`
`ScSetScannerAtoD`	`Control` routine with `csCode = 17`, and `csParam = Vrt (Byte)` and `csParam+2 = Vrb (Byte)`
`ScSetNoHome`	`Control` routine with `csCode = 6` and `csParam = noHome`
`ScSetSpeed`	`Control` routine with `csCode = 11` and `csParam = speed`
`ScSetThreshold`	`Control` routine with `csCode = 9` and `csParam = thresholdLevel`
`ScSetWaitButton`	`Control` routine with `csCode = 10` and `csParam = waitButton`
`ScVendorUnique`	`Control` routine with `csCode = 8192`, `csParam = paramType`, and `csParam + 2 = paramPtr`

SCSI Interface for Apple Scanners

If you are writing an application for a computer other than a Macintosh computer, you must use the Small Computer System Interface (SCSI), for which this chapter provides a general introduction. To communicate directly with the Apple scanners, you must understand how to use the SCSI commands, which are described in detail in Chapter 7.

If you are writing an application for the Macintosh computer, use the Apple scanner driver functions described in Chapters 2 through 5. There is no advantage, in this case, in sending commands directly to the scanner.

SCSI Hardware Interface

This section describes the interface presented by the SCSI hardware on the Apple scanners, including the supported bus phases. It also addresses unique SCSI features of the Apple Scanner, the OneScanner, and the Color OneScanner.

Connector assignments

Apple scanners incorporate a standard SCSI interface, using drivers and receivers that can support a maximum cable length of 6 meters. Table 6-1 lists pin assignments for the 50-pin SCSI connector.

Table 6-1 SCSI connector pin assignments

Pin Number.	Signal	Description	Direction from host
1–12, 14–25, 35-37, 39, 40, 42	GND	Ground	
13	None	Open	
26	–DB(0)	Data bit 0	Input/output
27	–DB(1)	Data bit 1	Input/output
28	–DB(2)	Data bit 2	Input/output
29	–DB(3)	Data bit 3	Input/output
30	–DB(4)	Data bit 4	Input/output
31	–DB(5)	Data bit 5	Input/output
32	–DB(6)	Data bit 6	Input/output
33	–DB(7)	Data bit 7	Input/output
34	–DB(P)	Data parity bit	Input/output
38	–TERMPWR	Terminator power	Output
41	–ATN	Attention	Input

Table 6-1 SCSI connector pin assignments (continued)

Pin Number.	Signal	Description	Direction from host
43	–BSY	Busy	Input/output
44	–ACK	Acknowledge	Input
45	–RST	Reset	Input/output
46	–MSG	Message	Output
47	–SEL	Select	Input/output
48	–C/D	Control/data	Output
49	–REQ	Request	Output
50	–I/O	Input/output	Output

The signals are defined as follows:

–DB(0-7), –DB(P) Eight data signals –DB(0-7) and the parity bit, –DB(P). These signals form the SCSI data bus. –DB7 is the most significant bit and has the highest priority during the Arbitration phase (see "Bus Phases," later in this section). A data bit is defined as a 1 when the corresponding signal value is TRUE. Apple scanners present odd parity. Note that the parity bit is not valid during the Arbitration phase.

–TERMPWR Terminator power. The Apple scanners present terminator power with the following characteristics:

Voltage:	4.0 volts DC to 5.25 volts DC
Current:	800 milliampere (mA) minimum source drive capability 1.0 mA maximum sink capability

–ATN Attention. The host computer drives this signal low to indicate an Attention condition, informing the scanner that the host has a message ready. The scanner receives this message by performing a Message Out phase. The host may assert this condition at any time except during the Arbitration and Bus Free phases.

–BSY Busy. Any device on the SCSI bus may hold this signal active to indicate that it is using the bus. The Busy signals from all devices on the SCSI bus are combined, so that if any device holds the signal active all other devices can detect that the bus is in use.

–ACK Acknowledge. The host computer drives this signal to acknowledge receipt of data during a –REQ/–ACK data transfer handshake.

–RST Reset. The host computer drives this signal low to indicate a Reset condition. This condition takes precedence over all other phases and conditions. The scanner may trigger a Reset condition by asserting this signal for a minimum of 25 microseconds. The Reset signals from all devices on the SCSI bus are combined, so that if any device holds the signal active all other devices can detect the Reset condition.

–MSG	Message. The scanner drives this signal low during the Message phase. The scanner asserts the – C/D, –I/O, and –MSG signals during the –REQ/–ACK handshake for this phase. During the Message In phase, the scanner may send a message to the host. During the Message Out phase, the scanner may request that a message be sent from the host to the scanner. The scanner may invoke the Message Out phase at its convenience, in response to an Attention condition asserted by the host.
–SEL	Select. The host computer drives this signal low to select the scanner, or any other SCSI device on the bus.
–C/D	Control/Data. The scanner drives this signal low to indicate it is sending control information to the host. If this signal is high, the scanner is sending data.
–REQ	Request. The scanner drives this signal low to request a –REQ/–ACK data transfer handshake.
–I/O	Input/output. The scanner uses this signal to control the direction of data transfer on the data bus. If this signal is TRUE (low), the scanner is sending data to the host. If this signal is FALSE (high), the scanner is receiving data from the host. The direction of data flow is referenced to the host CPU.

Bus Phases

The SCSI interface supports the following eight bus phases:.

Bus Free	Indicates that no SCSI devices are using the bus and that the bus is available for other users. The scanner goes to the Bus Free phase within 1 millisecond of completing a SCSI transaction.
Arbitration	Allows a SCSI device to gain control of the bus. Note that either the host or the scanner may gain control of the bus.
Selection	Allows the host to select the scanner. The host selects the scanner by asserting the –SEL signal. The scanner responds to a selection request within 200 milliseconds.
Reselection	Allows the scanner to reconnect to the host. This phase allows the scanner to resume an operation that was previously interrupted.
Command	Allows the scanner to request command information from the host. See Chapter 7, “SCSI Commands for Apple Scanners,” for information on the SCSI commands supported by the Apple Scanner, the OneScanner, and the Color OneScanner.
Data	Allows data to flow between the host and the scanner. This phase comprises both the Data In and Data Out phases. The state of the -I/O signal indicates the direction of data flow. During the Data In phase, the scanner sends data to the host. During the Data Out phase, the host sends data to the scanner. See Chapter 7, “SCSI Commands for Apple Scanners,” for information on the data formats supported by the Apple Scanner, the OneScanner, and the Color OneScanner.
Status	Allows the scanner to send status information to the host. See “SCSI Status Codes,” later in this chapter, for information on the status codes sent by the Apple Scanner, the OneScanner, and the Color OneScanner.

Message — Allows message information to flow between the host and the scanner. This phase comprises both the Message In and Message Out phases. The state of the I/O signal indicates the direction of message flow. During the Message In phase, the scanner sends message information to the host. During the Message Out phase, the host sends message information to the scanner. Several messages may be exchanged in either phase. See "SCSI Messages," later in this chapter, for information on the messages supported by the Apple Scanner, the OneScanner, and the Color OneScanner.

The Command, Data, Status, and Message phases are referred to as the ***information transfer phases*** because they are used together to transfer data or control information between devices on the SCSI bus. The host and scanner communicate with defined commands, which are transmitted in packets. (See Chapter 7, "SCSI Commands for Apple Scanners.") These message exchanges are called ***transactions.***

The information transfer phases work as follows: after the host sends a command to the scanner, the transaction switches to the Data or Status phase, depending on the command issued. If the command specifies a data transfer, the transaction enters the Data phase and the host and scanner exchange data appropriate to the command. If the command does not have a Data phase, the transaction goes straight to the Status phase. The scanner then sends 1 byte of status information to the host (see "SCSI Status Codes," later in this chapter). After transmitting the status information, the transaction switches to the Message phase, where one or more messages may be exchanged between the scanner and the host. (See "SCSI Messages," later in this chapter.)

SCSI Implementation by Apple Scanners

This section discusses details of the SCSI implementation supported by the Apple scanners. The following paragraphs describe the SCSI processing associated with hardware resets, scanner start up when power is turned on, and error processing during data transfers.

The scanner invokes a hardware reset whenever the host asserts the -RST (Reset signal) or sends a BUS DEVICE RESET message to the scanner. The scanner immediately enters the Bus Free phase, cancels any scan that is in progress, and restores its internal variables and parameter settings to their default values.

When power to the scanner is turned on, the scanner sets its SCSI ***sense data*** to unit attention (Sense key value of $06). The scanner accepts only REQUEST SENSE and INQUIRY commands—it rejects any other commands with a CHECK CONDITION status. You must issue a REQUEST SENSE command to clear the unit attention sense data before issuing any other SCSI commands.

If the scanner detects an error in a command header or parameter block, it does not execute the command. The scanner sets its SCSI sense data to illegal request (Sense key value of $05) and returns a CHECK CONDITION status. Note that the scanner always reads the entire command header before returning any error status. If a command includes optional parameters, the scanner reads all of the parameter data before returning the CHECK CONDITION status.

SCSI Status Codes

Apple scanners return a status code after every SCSI command, during the status bus phase, as shown below.

GOOD ($00)	Indicates that the scanner has successfully completed the command.
CHECK CONDITION ($02)	Indicates that the scanner has encountered an error. You should issue the REQUEST SENSE command to retrieve the SCSI sense data from the scanner.
RESERVATION CONFLICT ($18)	The scanner returns this status whenever it receives a command from a device that has not reserved the scanner.

SCSI Messages

SCSI messages allow devices on the SCSI bus to manage their interactions. The following sections discuss the messages supported by each of the Apple scanners.

Apple Scanner Message

The Apple Scanner supports a single 1-byte message.

COMMAND COMPLETE ($00)	The Apple Scanner sends this message to indicate that it has finished processing a command and has returned a status code to the initiator. After sending this message, the scanner enters the Bus Free phase by releasing the -BSY line.

OneScanner and Color OneScanner Messages

The OneScanner and Color OneScanner support identical messages of one or more bytes. In addition, one or more messages may be sent during a single Message bus phase. However, no single message may be split across more than one Message phase. The OneScanner and Color OneScanner support the following messages:

COMMAND COMPLETE ($00)	The scanner sends this message to indicate that it has finished processing a command and has returned a status code to the initiator. After sending this message, the scanner enters the Bus Free phase by releasing the -BSY line.
DISCONNECT ($04)	The scanner sends this message to indicate that it is going to disconnect temporarily from the bus. The scanner reconnects later to complete the current operation. After sending this message, the scanner enters the Bus Free phase by releasing the -BSY line. Note that it is an error for the scanner to release -BSY without first issuing a COMMAND COMPLETE or DISCONNECT message.
ABORT ($06)	The host sends this message to the scanner to clear the current operation. The scanner goes to the Bus Free Phase.
MESSAGE REJECT ($07)	The scanner sends this message to reject a message it has received. After sending this message, the scanner enters the Command phase.
BUS DEVICE RESET ($0C)	The host sends this message to the scanner to clear all current commands and force a reset operation. After recognizing this message, the scanner goes to the Bus Free Phase.
IDENTIFY ($80 or $C0)	Either the host or the scanner sends one of these messages to establish a connection across the SCSI bus. The host always sends a message code of $C0, indicating that the scanner may terminate the connection by issuing a DISCONNECT message. When establishing a connection, the scanner always sends a message code of $80.

SCSI Commands for Apple Scanners

SCSI Commands for Apple Scanners

The SCSI commands recognized by the Apple Scanner, the OneScanner, and the Color OneScanner make up a subset of the ***American National Standards Institute (ANSI)*** SCSI command set. Table 7-1 lists the commands and hexadecimal operation code used by the Apple scanners. This chapter describes each command in the order shown in Table 7-1. It describes the command format first for the Apple Scanner, then for the OneScanner, and finally for the Color OneScanner.

Table 7-1 SCSI commands and operation codes

Command	Hex opcode
TEST UNIT READY	$00
REQUEST SENSE	$03
INQUIRY	$12
MODE SELECT	$15
RESERVE	$16
RELEASE	$17
MODE SENSE	$1A
SCAN	$1B
SEND DIAGNOSTIC	$1D
DEFINE WINDOW PARAMETERS	$24
GET WINDOW PARAMETERS	$25
READ	$28
SEND	$2A
OBJECT POSITION	$31
GET DATA STATUS	$34

TEST UNIT READY ($00)

The TEST UNIT READY command provides a means for the host computer to check whether the scanner is ready for normal operation. If the scanner is in a ready condition, the command reports a GOOD status. If the scanner is not ready, the command sets the SCSI sense data to the cause of the error, and the command is terminated with a CHECK CONDITION status. To retrieve the sense data, issue the REQUEST SENSE command (described later in this chapter). Figure 7-1 shows the format of the TEST UNIT READY command structure.

This command is not a request for self-testing. For more information about internal scanner tests, see "SEND DIAGNOSTIC ($1D)," later in this chapter.

Figure 7-1 The TEST UNIT READY command structure

Byte number	Bit number 7	6	5	4	3	2	1	0
0	($00) Operation code							
1	($00) Reserved							
2	($00) Reserved							
3	($00) Reserved							
4	($00) Reserved							
5	($00) Reserved							

FIELD DESCRIPTIONS

Here is a list of the fields used in this command structure:

Operation code
: The operation code for the TEST UNIT READY command is $00.

Reserved
: These fields are reserved for future expansion. Set them to 0.

REQUEST SENSE ($03)

The REQUEST SENSE command instructs the scanner to return error and status information. The SCSI sense data is valid after any operation that returns a CHECK CONDITION status. If another operation returns a CHECK CONDITION status before your program retrieves the current sense data, the scanner overwrites the old sense data with the new data.

Figure 7-2 shows the format of the REQUEST SENSE command. The returned information is sent to the host computer in a return structure appropriate to the type of attached scanner. The return structures for the Apple Scanner, OneScanner, and Color OneScanner are described later in this section.

Figure 7-2 The REQUEST SENSE command structure

Byte number	Bit number 7	6	5	4	3	2	1	0
0	($03) Operation code							
1	($00) Reserved							
2	($00) Reserved							
3	($00) Reserved							
4	Transfer length							
5	($00) Reserved							

FIELD DESCRIPTIONS

Here is a list of the fields used in this command structure:

Operation code

The operation code for the REQUEST SENSE command is $03.

Reserved These fields are reserved for future expansion. Set them to 0.

Transfer length

This field indicates the number of bytes of REQUEST SENSE data to be returned by the scanner. A value of 0 in this field indicates that no data is to be returned (this is not an error). Any other value indicates the maximum number of sense bytes to be returned. The scanner stops transferring data when it has returned the number of bytes specified by the Transfer length field, or when all available REQUEST SENSE data has been returned, whichever comes first.

REQUEST SENSE Return Structure Description for the Apple Scanner

The REQUEST SENSE command returns a structure containing the requested information. Figure 7-3 shows the format of the return structure for the Apple Scanner.

Figure 7-3 The REQUEST SENSE return structure for the Apple Scanner

Byte number	Bit number 7	6	5	4	3	2	1	0
0	($0) Rsvd	(7)			($00) Reserved			
1	($00) Reserved							
2	($0) Reserved				Sense key			
3-6	($00) Reserved							
7	Additional sense length							
8-17	($00) Reserved							
18	Dim light	No light	No home	No limit	Shade error	ROM error	RAM error	CPU error
19	DIPP error	DMA error	GA1 error	($0) Reserved				
20	Optical gain							
21	($00) Reserved							

Here is a list of the fields and bits defined in this return structure:

Reserved — These fields are reserved for future expansion. Set them to 0.

Sense key — Bits 0–3 of byte 2 contain the ***Sense key.*** The possible Sense key values are as follows:

- $0 — No sense information: there is no specific sense data to be returned to the host computer. This situation occurs after a command executes successfully.
- $2 — Not ready: the scanner cannot scan without some user intervention.
- $4 — Hardware error: the scanner detects a nonrecoverable hardware fault while executing a command or during a self-test; these errors are indicated by the setting of the hardware error flags in bytes 18 and 19 of the return data block.
- $5 — Illegal request: the scanner detects an illegal parameter in the command block; the scanner cannot perform a scan and cannot alter any parameter data within the scanner.

$6 Unit attention: the scanner has just been switched on or reset and cannot execute any command except REQUEST SENSE or INQUIRY.

$9 Vendor unique: the scanner detects a vendor unique error, which is identified by the setting of the Vendor unique error flags in byte 18 of the return data block.

Additional sense length

This field indicates the number of sense data bytes that follow. The value of this field does not include bytes 0–7 of the return structure. It reflects the maximum size possible for a returned structure. The scanner does not adjust the value of this field to accommodate the Transfer length parameter specified in your REQUEST SENSE command.

Hardware error flags

Bytes 18 and 19 in the return structure contain bit flags that indicate the specific nature of a hardware error. Flags that are set to 1 indicate a specific hardware error condition. If the returned Sense key field is set to $4, then one or more of these bits are set to 1.

No light No light is detected from the fluorescent lamp.

No limit The scanner was unable to locate the limit switch position. Scanning cannot proceed because physical damage to the device may occur.

ROM error A fatal hardware error has been detected in the ROM (an erroneous checksum).

RAM error A fatal hardware error has been detected in the RAM.

CPU error A fatal hardware error has been detected in the CPU.

DIPP error A fatal hardware error has been detected in the document image preprocessor (DIPP) IC.

DMA error A fatal hardware error has been detected in the direct memory access (DMA) IC.

GA1 error A fatal hardware error has been detected in the Gate Array 1 (GA1) IC.

Vendor unique error flags

Byte 18 in the return structure contains bit flags that indicate the specific nature of a vendor unique error. Flags that are set to 1 indicate a specific error condition. If the returned Sense key field is set to $9, then one or more of these bits is set to 1.

Dim light The fluorescent lamp is functioning, but its output has dropped below 70 percent and is too low for a scan to be accurate.

No home The scanner was unable to locate the home position mark. Scanning may proceed, but the document may not be scanned properly.

Shade error — Three unsuccessful attempts were made to calibrate the charge-coupled device (CCD) array correctly. Scanning may continue, but the results may be poor. The cause may be a dim lamp or a component that needs servicing.

Optical gain — Indicates the relative intensity of the lamp by returning the gain by which the optical system must boost the analog amplifier to give correct shading results. (This condition is not an error and does not set the Sense key).

REQUEST SENSE Return Structure Description for the OneScanner

The REQUEST SENSE command returns a structure containing the requested information. Figure 7-4 shows the format of the return structure for the OneScanner.

Figure 7-4 The REQUEST SENSE return structure for the OneScanner

Byte number	Bit number 7	6	5	4	3	2	1	0
0	($0) Rsvd	(7)			($0) Reserved			
1	($00) Reserved							
2	($00) Reserved				Sense key			
3 – 6	($00) Reserved							
7	($0E) Additional sense length							
8 – 17	($00) Reserved							
18	Dim light	No light	No home	No limit	CCDCG	LRAM	CRAM	ROM
19	($0) Reserved				SRAM	CPU	($0) Reserved	
20	Optical gain							
21	($00) Reserved							

Here is a list of the fields and bits defined in this return structure:

Reserved — These fields are reserved for future expansion. Set them to 0.

Sense key — Bits 0–3 of byte 2 contain the Sense key. The possible Sense key values are as follows:

- $0 — No sense information: there is no specific sense information to be returned to the host computer. This situation occurs after a command has been executed successfully.
- $2 — Not ready: the scanner cannot scan without some user intervention.

$4 Hardware error: the scanner detects a nonrecoverable hardware fault while executing a command or during a self-test; these errors are indicated by the setting of the hardware error flags in bytes 18 and 19 of the return data block.

$5 Illegal request: the scanner detects an illegal parameter in the command block; the scanner cannot perform a scan and cannot alter any parameter data within the scanner.

$6 Unit attention: the scanner has just been switched on or reset and cannot execute any command except REQUEST SENSE or INQUIRY.

$9 Vendor unique: the scanner detects a vendor unique error, which is identified by the setting of the vendor unique error flag in byte 18 of the return data block.

Additional sense length

The value in this field indicates the number of sense data bytes that follow. The value does not include bytes 0–7 of the return structure. It reflects the maximum size possible for a returned structure. The scanner does not adjust the value of this field to accommodate the Transfer length parameter specified in your REQUEST SENSE command.

Hardware error flags

Bytes 18 and 19 in the return structure contain bit flags that indicate the specific nature of a hardware error. Flags that are set to 1 indicate a specific hardware error condition. If the returned Sense key field is set to $4, then one or more of these bits are set to 1.

No light No light is detected from the fluorescent lamp.

No home The scanner was unable to locate the home position mark. Scanning may proceed, but the document may not be scanned properly.

No limit The scanner was unable to locate the limit switch position. Scanning cannot proceed because physical damage to the device may occur.

CCDCG CCD clock generator is not functioning properly. Unit should be repaired.

LRAM Line RAM has one or more bad bits. Unit should be repaired.

CRAM Correction RAM has one or more bad bits. Unit should be repaired.

ROM Bad checksum in the control ROM. Unit should be repaired.

SRAM General-purpose RAM for the scanner CPU has one or more bad bits. Unit should be repaired.

CPU CPU is not functioning properly. Unit should be repaired.

Vendor unique error flag

Bit 7 of byte 18 in the return structure contains a bit flag that indicates the specific nature of a vendor unique error, in this case, Dim light. If the flag is set, it indicates the specific error condition.

Dim light — The fluorescent lamp is functioning, but its output has dropped below 70 percent and is too low for a scan to be accurate.

Optical gain — Indicates the relative intensity of the lamp by returning the gain by which the optical system must boost the analog amplifier to give correct shading results. (This condition is not an error and does not set the Sense key).

REQUEST SENSE Return Structure Description for the Color OneScanner

The REQUEST SENSE command returns a structure containing the requested information. Figure 7-5 shows the format of the return structure for the Color OneScanner.

Figure 7-5 The REQUEST SENSE return structure for the Color OneScanner

Byte number	Bit number 7	6	5	4	3	2	1	0
0	($0) Rsvd	($70) Current error			($00) Reserved			
1	($00) Reserved							
2	($0) Reserved				Sense key			
3 – 6	($00) Reserved							
7	($0E) Additional sense length							
8 – 17	($00) Reserved							
18	Dim light	No light	No home	No limit	Cal Ckt	PSRAM	SRAM	ROM
19	Rsvd	Rsvd	Rsvd	ICP	Rsvd	Rsvd	Over light	Rsvd
20	Optical gain							
21	($00) Reserved							

Here is a list of the fields and bits defined in this return structure:

Reserved — These fields are reserved for future expansion. Set them to 0.

Current error

This field indicates a current scanner error.

Sense key Bits 0–3 of byte 2 contain the Sense key. The possible Sense key values are as follows:

- $0 No sense information: there is no specific sense information to be returned to the host computer. This situation occurs after a command has been executed successfully.
- $2 Not ready: the scanner cannot perform a scan because it is already busy scanning or because there is a problem not covered by the other Sense keys. The operator may have to intervene to correct the problem.
- $4 Hardware error: the scanner detects a nonrecoverable hardware fault while executing a command or during a self-test; these errors are indicated by the setting of the hardware error flags in bytes 18 and 19 of the return data block.
- $5 Illegal request: the scanner detects an illegal parameter in the command block; the scanner cannot perform a scan and cannot alter any parameter data within the scanner.
- $6 Unit attention: the scanner has just been switched on or reset and cannot execute any command except REQUEST SENSE or INQUIRY.
- $9 Vendor unique: the scanner detects a vendor-unique error, which is identified by the setting of the vendor unique error flags in bytes 18 and 19 of the return data block.

Additional sense length

This field indicates the number of sense data bytes that follow. The value of this field does not include bytes 0–7 of the return structure. It reflects the maximum size possible for a returned structure. The scanner does not adjust the value of this field to accommodate the Transfer length parameter specified in the REQUEST SENSE command.

Hardware error flags

Bytes 18 and 19 in the return structure contain bit flags that indicate the specific nature of a hardware error. Flags that are set to 1 indicate a specific hardware error condition. If the returned Sense key field is set to $4, then one or more of these bits are set to 1.

- No light — No light is detected from the fluorescent lamp.
- No limit — The scanner was unable to locate the limit switch position. Scanning cannot proceed because physical damage to the device may occur.
- Cal Ckt — The calibration circuit cannot support normal shading correction. Scanning should not proceed.
- PSRAM — The correction RAM has one or more bad bits. Scanning should not proceed.
- SRAM — General-purpose RAM for the scanner CPU has one or more bad bits. Unit should be repaired.
- ROM — Bad checksum in the control ROM. Unit should be repaired.

ICP — The ICP (image correction processor) is not functioning properly after a read/write test of the ICP registers. Sense key $4 should be set when the ICP error bit is set to 1. Scanning should not proceed.

Vendor-unique error flags

Bytes 18 and 19 in the return structure contain bit flags that indicate the specific nature of a vendor unique error. Flags that are set to 1 indicate a specific error condition. If the returned Sense key field is set to $9, then one or more of these bits are set to 1.

Dim light — The fluorescent lamp is functioning, but its output has dropped below 70 percent and is too low for a scan to be accurate.

Lamp needs replacing.

The white target underneath the scanning bed is dirty. (Color OneScanner only)

The three mirrors are dirty. (Color OneScanner only)

The aperture plate is dirty. (Color OneScanner only)

The CCD is dirty or bad. (Color OneScanner only)

Scanning may continue, but results may not be satisfactory.

No home — The scanner was unable to locate the home position mark. Scanning may proceed, but the document may not be scanned properly.

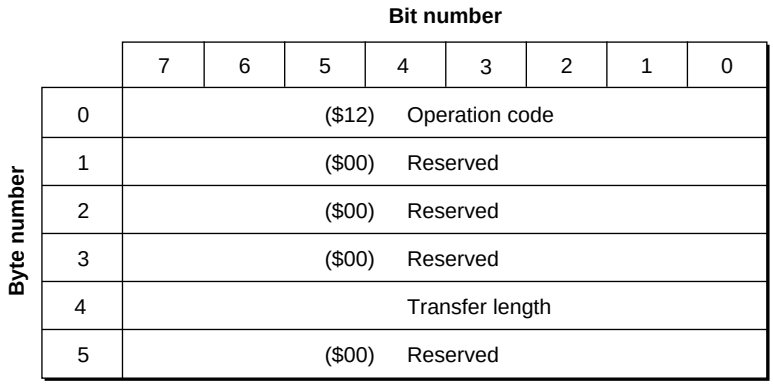
Over light — The fluorescent lamp is too bright for normal scanning. Scanning can proceed, but the document may not be scanned properly.

Optical gain — Indicates the relative intensity of the lamp by returning the gain by which the optical system must boost the analog amplifier to give correct shading results. (This condition is not an error and does not set the Sense key.)

INQUIRY ($12)

The INQUIRY command returns information about the scanner's device type, firmware version, conformance to various industry standards, and support for the SCSI command set. Figure 7-6 shows the format of the INQUIRY command structure. The scanner returns the requested information in a return structure appropriate to the scanner type. These return structures are described later in this section.

Figure 7-6 The INQUIRY command structure



FIELD DESCRIPTIONS

Here is a list of the fields used in this command structure:

Operation code
: The operation code for the INQUIRY command is $12.

Reserved
: These fields are reserved for future expansion. Set them to 0.

Transfer length
: This field indicates the number of bytes of INQUIRY data to be returned by the scanner. A value of 0 in the transfer length field indicates that no data is to be returned (this is not an error). Any other value indicates the maximum number of data bytes to be returned. The scanner stops transferring data when it has returned the number of bytes specified by the Transfer length field, or all available INQUIRY data, whichever comes first.

INQUIRY Return Structure Description for the Apple Scanner

The INQUIRY return structure for the Apple Scanner contains a 5-byte header followed by 44 bytes of additional inquiry information, as shown in Figure 7-7.

Figure 7-7 The INQUIRY return structure for the Apple Scanner

Byte number	Bit number 7	6	5	4	3	2	1	0
0	($06) Device type							
1	($00) Reserved							
2	Version							
3	($00) Reserved				Response data format			
4	($2C) Additional length							
5 – 7	($00) Reserved							
8 – 15	Vendor identification							
16 – 31	Product identification							
32 – 35	Revision level							
36	($00) Reserved							
37	($00) Reserved							
38	($00) Group 0 commands opcode							
39	($90) Enabled opcodes: $00, $03							
40	($00) Enabled opcodes: none							
41	($27) Enabled opcodes: $12, $15, $16, $17							
42	($34) Enabled opcodes: $1A, $1B, $1D							
43	($01) Group 1 commands opcode							
44	($08) Enabled opcodes: $24							
45	($A0) Enabled opcodes: $28, $2A							
46	($08) Enabled opcodes: $34							
47	($00) Enabled opcodes: none							
48	($FF) End of block							

Here is a list of the fields defined in this return structure:

Device type This field contains the SCSI standard device code of the scanner, which is $06.

Reserved These fields are reserved for future expansion. Set them to 0.

Version This field indicates the version number of this scanner, which is $02, as specified in the ANSI SCSI-2 (X3T9.2/86-109) device interface specification.

Response data format
This field indicates that the inquiry data format is as specified in the ANSI SCSI-2 (X3T9.2/86-109) specification. This field is $02 for the Apple Scanner.

Additional length
This field indicates the number of data bytes that follow. The value of this field is $2C and does not include bytes 0–4 of the return structure. The scanner does not adjust the value of this field to accommodate the Transfer length field specified in your INQUIRY command.

Vendor identification
This field contains an ASCII string that identifies the scanner's vendor. For the Apple Scanner, this string is "`APPLE`", followed by three spaces.

Product identification
This field contains an ASCII string identifying the scanner's product name and a model number. For the Apple Scanner, this string is "`SCANNER A9M0337`", followed by one space.

Revision level
This field contains an ASCII string that identifies the scanner's firmware revision level. For the Apple Scanner, this string is "`0.00`".

Group 0 commands opcode
A value of 0 in this field indicates that at least one of the Group 0 scanning command operation codes (as defined in the ANSI SCSI-2 specification) is supported. Bytes 39–42 contain bit flags indicating the Group 0 commands supported by the Apple Scanner. Each command has a corresponding bit in the appropriate bit flag. That bit is set to 1 if the scanner supports the command, and it is set to 0 if the scanner does not support the command.

Enabled opcodes ($00 through $07)
These bit flags indicate the SCSI commands from $00 to $07 that the Apple Scanner supports. The Apple Scanner supports commands $00 (TEST UNIT READY) and $03 (INQUIRY). Therefore, bits 7 and 4 are set to 1. All other bits are set to 0. The value of the field is $90.

Enabled opcodes ($08 through $0F)
None of these operations is supported; all bits are set to 0.

Enabled opcodes ($10 through $17)

These bit flags indicate the SCSI commands from $10 to $17 that the Apple Scanner supports. The Apple Scanner supports commands $12 (INQUIRY), $15 (MODE SELECT), $16 (RESERVE), and $17 (RELEASE). Therefore, bits 5, 2, 1, and 0 are set to 1. All other bits are set to 0. The value of the field is $27.

Enabled opcodes ($18 through $1F)

These bit flags indicate the SCSI commands from $18 to $1F that the Apple Scanner supports. The Apple Scanner supports commands $1A (MODE SENSE), $1B (SCAN), and $1D (SEND DIAGNOSTIC). Therefore, bits 5, 4, and 2 are set to 1. All other bits are set to 0. The value of the field is $34.

Group 1 commands

A value of 1 in this field indicates that at least one of the Group 1 scanning commands (as defined in the ANSI SCSI-2 specification) is supported. Bytes 44–47 contain bit flags indicating the Group 1 commands supported by the Apple Scanner. Each command has a corresponding bit in the appropriate bit flag. That bit is set to 1 if the scanner supports the command, and it is set to 0 if the scanner does not support the command.

Enabled opcodes ($20 through $27)

These bit flags indicate the SCSI commands from $20 to $27 that the Apple Scanner supports. The Apple Scanner supports command $24 (DEFINE WINDOW PARAMETERS). Therefore, bit 3 is set to 1. All other bits are set to 0. The value of the field is $08.

Enabled opcodes ($28 through $2F)

These bit flags indicate the SCSI commands from $28 to $2F that the Apple Scanner supports. The Apple Scanner supports commands $28 (READ) and $2A (SEND). Therefore, bits 7 and 5 are set to 1. All other bits are set to 0. The value of the field is $A0.

Enabled opcodes ($30 through $37)

These bit flags indicate the SCSI commands from $30 to $37 that the Apple Scanner supports. The Apple Scanner supports command $34 (GET DATA STATUS). Therefore, bit 3 is set to 1. All other bits are set to 0. The value of the field is $08.

Enabled opcodes ($38 through $3F)

None of these operation codes is supported; all bits are set to 0.

End of block A value of $FF in this field is used to indicate the end of the block.

INQUIRY Return Structure Description for the OneScanner

The INQUIRY return structure for the OneScanner contains a 5-byte header followed by 44 bytes of additional inquiry information, as shown in Figure 7-8.

Here is a list of the fields defined in this return structure:

Device type — This field contains the SCSI standard device code of the scanner, which is $06.

Reserved — These fields are reserved for future expansion. Set them to 0.

Version — This field indicates the version number of this scanner, which is $02, as specified in the ANSI SCSI-2 (X3T9.2/86-109) device interface specification.

Response data format — This field indicates that the inquiry data format is as specified in the ANSI SCSI-2 (X3T9.2/86-109) specification. This field is $02 for the OneScanner.

Additional length — This field indicates the number of data bytes that follow. The value of this field is $2C and does not include bytes 0–4 of the return structure. The scanner does not adjust the value of this field to accommodate the Transfer length specified in your INQUIRY command.

ROM size — This field indicates the size, in kilobytes, of the scanner ROM. The value of this field is $20.

CRAM size — This field indicates the size, in kilobytes, of the correction RAM. The value of this field is $04.

SRAM size — This field indicates the size, in kilobytes, of the general storage RAM for the scanner CPU. The value of this field is $08.

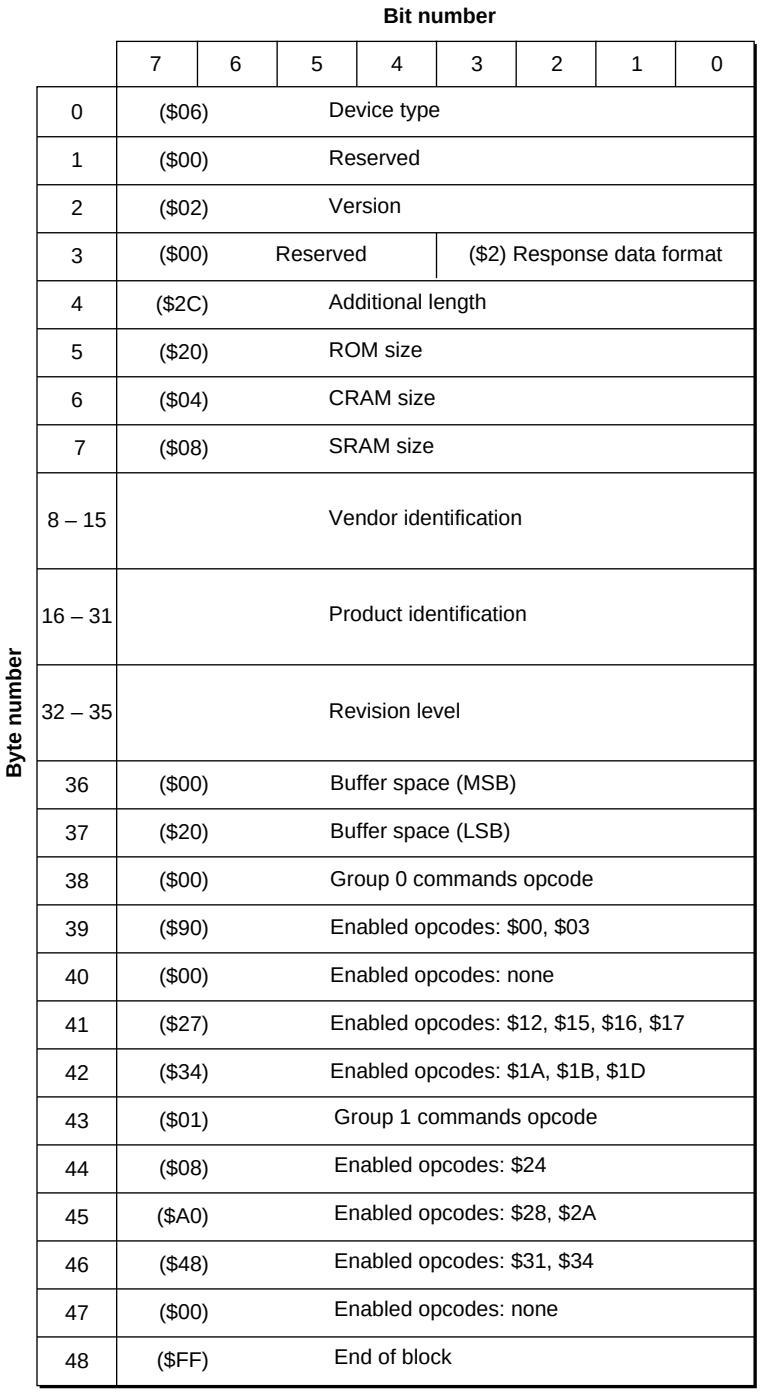
Vendor identification — This field contains an ASCII string that identifies the scanner's vendor. For the OneScanner, this string is `"APPLE"`, followed by three spaces.

Product identification — This field contains an ASCII string identifying the scanner's product name and a model number. For the OneScanner, this string is `"SCANNER II"`, followed by six spaces.

Revision level — This field contains an ASCII string that identifies the scanner's firmware revision level. For the OneScanner, this string is `"2.02"`.

Buffer space — Bytes 36 and 37 indicate the total amount of buffer memory, in kilobytes, in the attached scanner. Buffer memory is used to store scan image data until the host computer reads the data from the scanner. Together, these two bytes contain a 16-bit value: byte 36 contains the most significant byte; byte 37 contains the least significant byte. The value of byte 36 is $00; the value of byte 37 is $20.

Figure 7-8 The INQUIRY return structure for the OneScanner



Group 0 commands opcode

A value of 0 in this field indicates that at least one of the Group 0 scanning command operation codes (as defined in the ANSI SCSI-2 specification) is supported. Bytes 39–42 contain bit flags indicating the Group 0 commands supported by the OneScanner. Each command has a corresponding bit in the appropriate bit flag. That bit is set to 1 if the OneScanner supports the command, and it is set to 0 if the OneScanner does not support the command.

Enabled opcodes ($00 through $07)

These bit flags indicate the SCSI commands from $00 to $07 that the OneScanner supports. The OneScanner supports commands $00 (TEST UNIT READY) and $03 (REQUEST SENSE). Therefore, bits 7 and 4 are set to 1. All other bits are set to 0. The value of the field is $90.

Enabled opcodes ($08 through $0F)

None of these operations is supported; all bits are set to 0.

Enabled opcodes ($10 through $17)

These bit flags indicate the SCSI commands from $10 to $17 that the OneScanner supports. The OneScanner supports commands $12 (INQUIRY), $15 (MODE SELECT), $16 (RESERVE), and $17 (RELEASE). Therefore, bits 5, 2, 1, and 0 are set to 1. All other bits are set to 0. The value of the field is $27.

Enabled opcodes ($18 through $1F)

These bit flags indicate the SCSI commands from $18 to $1F that the OneScanner supports. The OneScanner supports commands $1A (MODE SENSE), $1B (SCAN), and $1D (SEND DIAGNOSTIC). Therefore, bits 5, 4, and 2 are set to 1. All other bits are set to 0. The value of the field is $34.

Group 1 commands

A value of 1 in this field indicates that at least one of the Group 1 scanning commands (as defined in the ANSI SCSI-2 specification) is supported. Bytes 44–47 contain bit flags indicating the Group 1 commands supported by the OneScanner. Each command has a corresponding bit in the appropriate bit flag. That bit is set to 1 if the OneScanner supports the command, and it is set to 0 if the OneScanner does not support the command.

Enabled opcodes ($20 through $27)

These bit flags indicate the SCSI commands from $20 to $27 that the OneScanner supports. The OneScanner supports command $24 (DEFINE WINDOW PARAMETERS). Therefore, bit 3 is set to 1. All other bits are set to 0. The value of the field is $08.

Enabled opcodes ($28 through $2F)

These bit flags indicate the SCSI commands from $28 to $2F that the OneScanner supports. The OneScanner supports commands $28 (READ) and $2A (SEND). Therefore, bits 7 and 5 are set to 1. All other bits are set to 0. The value of the field is $A0.

Enabled opcodes ($30 through $37)
: These bit flags indicate the SCSI commands from $30 to $37 that the OneScanner supports. The OneScanner supports commands $31 (OBJECT POSITION) and $34 (GET DATA STATUS). Therefore, bits 6 and 3 are set to 1. All other bits are set to 0. The value of the field is $48.

Enabled opcodes ($38 through $3F)
: None of these operations is supported; all bits are set to 0.

End of block
: A value of $FF in this field is used to indicate the end of the block.

INQUIRY Return Structure Description for the Color OneScanner

The INQUIRY return structure for the Color OneScanner contains a 5-byte header followed by 48 bytes of additional inquiry information, as shown in Figure 7-9 on page 130.

Here is a list of the fields defined in this return structure:

Device type
: This field contains the SCSI standard device code of the scanner, which is $06.

Reserved
: These fields are reserved for future expansion. Set them to 0.

Version
: This field indicates the version number of this scanner, which is $02, as specified in the ANSI SCSI-2 (X3T9.2/86-109) device interface specification.

Response data format
: This field indicates that the inquiry data format is as specified in the ANSI SCSI-2 (X3T9.2/86-109) specification. The value of this field is $02 for the Color OneScanner.

Additional length
: This field indicates the number of data bytes that follow. The value of this field is $30 and does not include bytes 0–4 of the return structure. The scanner does not adjust the value of this field to accommodate the Transfer length specified in your INQUIRY command.

Vendor identification
: This field contains an ASCII string that identifies the scanner's vendor. For the Color OneScanner, this string is `"APPLE"`, followed by three spaces.

Product identification
: This field contains an ASCII string identifying the scanner's product name and a model number. For the Color OneScanner, this string is `"SCANNER III"`, followed by five spaces.

Revision level
: This field contains an ASCII string that identifies the scanner's firmware revision level. For the Color OneScanner, this string is `"3.00"`.

Figure 7-9 The INQUIRY return structure for the Color OneScanner

Byte number	Bit number 7	6	5	4	3	2	1	0
0	($06) Device type							
1	($00) Reserved							
2	($02) Version							
3	($00) Reserved				($2) Response data format			
4	($30) Additional length							
5-7	($00) Reserved							
8-15	Vendor identification							
16-31	Product identification							
32-35	Revision level							
36	($00) Buffer space (MSB)							
37	($80) Buffer space (LSB)							
38	($00) Group 0 commands							
39	($90) Enabled opcodes: $00, $03							
40	($00) Enabled opcodes: none							
41	($27) Enabled opcodes: $12, $15, $16, $17							
42	($34) Enabled opcodes: $1A, $1B, $1D							
43	($01) Group 1 commands opcode							
44	($0C) Enabled opcodes: $24, $25							
45	($A0) Enabled opcodes: $28, $2A							
46	($48) Enabled opcodes: $31, $34							
47	($00) Enabled opcodes: none							
48	($FF) End of block							
49	($40) ROM size							
50	($01) PSRAM size (MSB)							
51	($00) PSRAM size (LSB)							
52	($08) SRAM size							

ROM size This field indicates the size, in kilobytes, of the scanner ROM. The value of this field is $40, equivalent to 64K decimal.

SRAM size This field indicates the size, in kilobytes, of the general storage RAM for the scanner CPU. The value of this field is $08.

PSRAM size This field is a two-byte field that indicates the size, in kilobytes, of the correction RAM. It contains $01 and $00, which are equivalent to 256K decimal.

Buffer space Bytes 36 and 37 indicate the total amount of buffer memory, in kilobytes, in the attached scanner. Buffer memory is used to store scan image data until the host computer reads the data from the scanner. Together, these two bytes contain a 16-bit value: byte 36 contains the most significant byte; byte 37 contains the least significant byte. The value of byte 36 is $00; the value of byte 37 is $80.

Group 0 commands opcode
A 0 in this field indicates that at least one of the Group 0 scanning command operation codes (as defined in the ANSI SCSI-2 specification) is supported. Bytes 39–52 contain bit flags indicating the Group 0 commands supported by the Color OneScanner. Each command has a corresponding bit in the appropriate bit flag. That bit is set to 1 if the Color OneScanner supports the command, and it is set to 0 if the Color OneScanner does not support the command.

Enabled opcodes ($00 through $07)
These bit flags indicate the SCSI commands from $00 to $07 that the Color OneScanner supports. The Color OneScanner supports commands $00 (TEST UNIT READY) and $03 (INQUIRY). Therefore, bits 7 and 4 are set to 1. All other bits are set to 0. The value of the field is $90.

Enabled opcodes ($08 through $0F)
None of these operation codes is supported; all bits are set to 0.

Enabled opcodes ($10 through $17)
These bit flags indicate the SCSI commands from $10 to $17 that the Color OneScanner supports. The Color OneScanner supports commands $12 (INQUIRY), $15 (MODE SELECT), $16 (RESERVE), and $17 (RELEASE). Therefore, bits 5, 2, 1, and 0 are set to 1. All other bits are set to 0. The value of the field is $27.

Enabled opcodes ($18 through $1F)
These bit flags indicate the SCSI commands from $18 to $1F that the Color OneScanner supports. The Color OneScanner supports commands $1A (MODE SENSE), $1B (SCAN), and $1D (SEND DIAGNOSTIC). Therefore, bits 5, 4, and 2 are set to 1. All other bits are set to 0. The value of the field is $34.

Group 1 commands

A value of 1 in this field indicates that at least one of the Group 1 scanning commands (as defined in the ANSI SCSI-2 specification) is supported. Bytes 44–47 contain bit flags indicating the Group 1 commands supported by the Color OneScanner. Each command has a corresponding bit in the appropriate bit flag. That bit is set to 1 if the Color OneScanner supports the command, and it is set to 0 if the Color OneScanner does not support the command.

Enabled opcodes ($20 through $27)

These bit flags indicate the SCSI commands from $20 to $27 that the Color OneScanner supports. The Color OneScanner supports commands $24 (DEFINE WINDOW PARAMETERS) and $25 (GET WINDOW PARAMETERS). Therefore, bits 2 and 3 are set to 1. All other bits are set to 0. The value of the field is $0C.

Enabled opcodes ($28 through $2F)

These bit flags indicate the SCSI commands from $28 to $2F that the Color OneScanner supports. The Color OneScanner supports commands $28 (READ) and $2A (SEND). Therefore, bits 7 and 5 are set to 1. All other bits are set to 0. The value of the field is $A0.

Enabled opcodes ($30 through $37)

These bit flags indicate the SCSI commands from $30 to $37 that the Color OneScanner supports. The Color OneScanner supports commands $31 (OBJECT POSITION) and $34 (GET DATA STATUS). Therefore, bits 6 and 3 are set to 1. All other bits are set to 0. The value of the field is $48.

Enabled opcodes ($38 through $3F)

None of these operation codes is supported; all bits are set to 0.

End of block A value of $FF in this field is used to indicate the end of the block.

MODE SELECT ($15)

The MODE SELECT command passes a parameter block to the scanner. The scanner uses this data to configure its operating parameters. Figure 7-10 shows the format of the MODE SELECT command structure. The information is passed to the scanner in a MODE SELECT parameter list that is appropriate to the parameter type and attached scanner. These parameter lists are described later in this section.

Figure 7-10 The MODE SELECT command structure

Byte number \ Bit number	7	6	5	4	3	2	1	0
0	($15)			Operation code				
1	($00)	Reserved		Page format	($00)		Reserved	
2	($00)			Reserved				
3	($00)			Reserved				
4	Parameter list length							
5	($00)			Reserved				

FIELD DESCRIPTIONS

Here is a list of the fields and bits used in this command structure:

Operation code
: The operation code for this command is $15.

Reserved
: These fields are reserved for future expansion. Set them to 0.

Page format
: The Page format bit indicates that all parameters after the block descriptors are unique to the vendor, as specified in this reference. This bit must always be set to 1.

Parameter list length
: This field contains the length, in bytes, of the parameter list to be passed to the scanner. A transfer length of 0 indicates that no parameter list is being passed (this is not an error condition).

MODE SELECT Parameter List Description

The MODE SELECT parameter list contains one or more optional parameter pages. The parameter page may be either an Apple-specific parameter page or a disconnect-reconnect parameter page. Each parameter page includes 2 bytes that specify the type and the length of the data page. The data pages are described next.

Apple-Specific Parameter Page Description for the Apple Scanner

Figure 7-11 shows the Apple-specific parameter page for the Apple Scanner.

Figure 7-11 The MODE SELECT Apple-specific parameter page for the Apple Scanner

Byte number	Bit number 7	6	5	4	3	2	1	0
0 - 3	($00) Reserved							
4	($00) Page code							
5	($6) Page length							
6	($0) Reserved							Graymap
7	Auto background-adjustment threshold							
8	($00) Reserved							Lamp
9	($00) Reserved							
10	($00) Reserved							
11	($00) Reserved							

Here is a list of the fields and bits used in this page:

Reserved — These fields are reserved for future expansion. Set them to 0.

Page code — This field identifies the page type. Set them to 0.

Page length — This field contains the length, in bytes, of the data portion of the parameter page, not including bytes 0–5. Set this field to 6.

Graymap — This field contains the value that controls the graymap function of the scanner. The graymap function allows the scanner to enhance detail in light areas or dark areas of a document, or not to enhance either. A value of 0 results in more detail in the darker areas of the document. A value of 1 results in no alteration of the data. A value of 2 results in more detail in the lighter areas of the document.

Auto background-adjustment threshold — This field specifies the threshold level at which the scanner determines whether a dot is black or white when the user selects automatic background adjustment. The default threshold level is 64.

When excessively dark areas of an original document are scanned, automatic background adjustment constantly adjusts the brightness level, resulting in more detail in the darker areas. Light areas are left unaltered.

To select automatic background adjustment, set the Threshold parameter to 0 in the DEFINE WINDOW PARAMETERS command parameter list. See "DEFINE WINDOW PARAMETERS ($24)," later in this chapter.

Lamp — This bit controls the fluorescent lamp used to illuminate the document during scanning. A value of 1 turns on the lamp. A value of 0 turns it off.

In addition, other conditions may affect whether the lamp is on or off. Once the lamp is turned on, it remains on until two minutes elapse without SCSI activity. The next SCSI command turns on the lamp for a minimum period of two minutes. Furthermore, if the No home bit is set to 1 in the SCAN command, the scanner turns off the lamp only after the carriage assembly returns to the home position. See "SCAN ($1B)," later in this chapter.

Apple-Specific Parameter Page Description for the OneScanner

Figure 7-12 shows the Apple-specific parameter page for the OneScanner.

Figure 7-12 The MODE SELECT Apple-specific parameter page for the OneScanner

Byte number	Bit number 7	6	5	4	3	2	1	0
0 - 3	($00)				Reserved			
4	($00)				Page code			
5	($06)				Page length			
6	($00)				Reserved			
7	($0)		Reserved			LED	Reset	Button
8	($0)		Reserved		Fast H	Fast L	CCD	Lamp
9	($00)				Reserved			
10	($00)				Reserved			
11	($00)				Reserved			

Here is a list of the fields and bits used in this page:

Reserved — These fields are reserved for future expansion. Set them to 0.

Page code — This field identifies the page type. Set it to 0.

Page length — This field contains the length, in bytes, of the data portion of the parameter page, not including bytes 0–5. Set it to $06.

LED — This bit controls the setting of the amber LED on the OneScanner. A value of 1 turns on the LED. A value of 0 turns it off.

Reset — This bit indicates whether the scanner should reset the Button bit. A value of 1 causes the scanner to set the Button bit to 0. A value of 0 has no effect.

Button — This bit indicates the current state of the button on the OneScanner since the last reset request (a MODE SELECT command with the Reset bit set to 1). A value of 1 indicates that the button has been pressed. A value of 0 indicates that the button has not been pressed. Applies only to scanners with ROM version 2.03, or earlier.

Fast H, Fast L — Together, these bits determine the scan speed for the scanner. Here are the three possible values for these bits:

Fast H	Fast L	Scan speed
0	0	Normal speed
0	1	High speed
1	0	Fast speed
1	1	Invalid value

At normal speed, the scanner processes the image at the slowest possible carriage speed and performs handshaking on all data exchanges with the host computer. At high speed, the scanner runs the carriage at high speed but still performs handshaking on all data exchanges with the host computer. At normal and high speeds, data handshaking prevents the loss of image data.

At fast speed, the scanner runs the carriage at high speed but does not perform any handshaking during data exchanges with the host. Eliminating the handshaking greatly improves the data exchange rate between the scanner and the host computer. However, image data may be lost if the host computer cannot keep up with the scanner. If the scanner loses data due to an overrun condition, it aborts the scan with a CHECK CONDITION status.

CCD — This bit controls power to the CCD array in the scanner. A value of 1 causes the scanner to switch on the CCD array whenever the scanner lamp is turned on. A value of 0 causes the scanner to switch on the CCD array only in response to SCAN operations.

Lamp — This bit controls the fluorescent lamp used to illuminate the document during scanning. A value of 1 turns on the lamp. A value of 0 turns it off.

In addition, other conditions may affect whether the lamp is on or off. Once the lamp has been turned on, it remains on until two minutes elapse without SCSI activity. The next SCSI command turns on the lamp for a minimum period of two minutes.

Apple-Specific Parameter Page Description for the Color OneScanner

Figure 7-13 shows the Apple-specific parameter page for the Color OneScanner.

Figure 7-13 The MODE SELECT Apple-specific parameter page for the Color OneScanner

Byte number	Bit number 7	6	5	4	3	2	1	0
0-3	($00) Reserved							
4	($0) Rsvd		($0) Apple unique control parameters					
5	($06) Parameter length							
6	($00) Reserved							
3	($0) Reserved					Scan LED	Gamma reset	3x3 reset
8	($0) Reserved			MTF circuit	ICP bypass	Data polarity	CCD on	Lamp on
9	Color sense select							
10	($00) Reserved							
11	($00) Reserved							

Here is a list of the fields and bits used in this page:

Reserved These fields are reserved for future expansion. Set them to 0.

Apple-unique control parameters

Apple unique control parameters are reserved for specific Apple functions. This field is reserved for Page Code, which identifies the page type. Set it to 0.

Parameter length

This field contains the length, in bytes, of the data portion of the parameter page, not including bytes 0-5. Set it to $06.

Scan LED This bit controls the setting of the amber LED on the Color OneScanner. A value of 1 turns on the LED. A value of 0 turns it off.

Gamma reset

When this bit is set (1) the scanner loads the gamma table with default values, which are chosen so that the output data is the same as the input data. The default value is 0.

3 x 3 reset This bit directs the scanner to load the default 3-by-3 color correction table. When this bit is set (1), the following default 3-by-3 matrix is loaded. The default value is 0.

1	0	0
0	1	0
0	0	1

MTF circuit Turns the MTF (modulation transfer function) peaking circuit on or off.

1	MTF peaking circuit on
0	MTF peaking circuit off

ICP bypass This bit decides whether the ICP is to be used or bypassed. When the field is set (1) the circuit is bypassed. When it is reset (0) the circuit is used.

Data polarity This bit determines the polarity of the output image data. When it is set (1) 00 is white and FF is black. In normal mode, it is reset to 0, and 00 is black and FF white.

CCD on This bit controls power to the CCD array in the scanner. A value of 1 causes the scanner to power on the CCD array whenever the scanner lamp is turned on. A value of 0 causes the scanner to power on the CCD array only in response to SCAN operations.

Lamp on This bit controls the fluorescent lamp used to illuminate the document during scanning. A value of 1 turns on the lamp. A value of 0 turns it off.

In addition, other conditions may affect whether the lamp is on or off. Once the lamp has been turned on, it remains on until two minutes elapse without SCSI activity. The next SCSI command turns on the lamp for a minimum period of two minutes.

Color sensor select

The Color sensor select field decides which color sensor line or lines are selected for scanning. The green line sensor is the default for gray scale scanning.

Disconnect-Reconnect Parameter Page Description

Figure 7-14 shows the disconnect-reconnect parameter page. You may use this page for the Apple Scanner, the OneScanner, and the Color OneScanner.

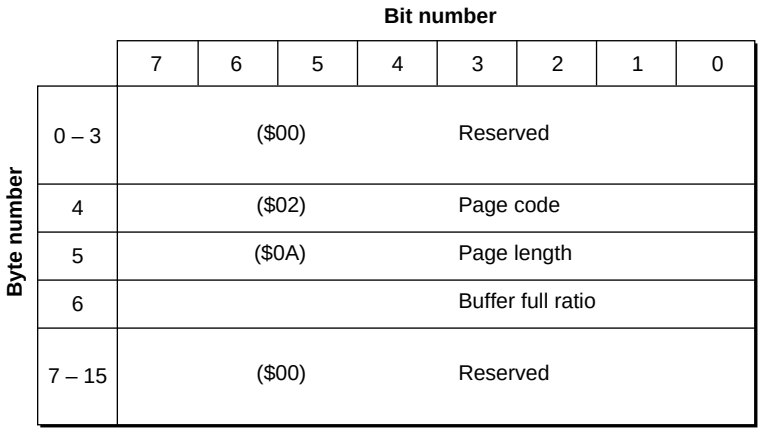
Here is a list of the fields used in this page:

Reserved These fields are reserved for future expansion. Set them to 0.

Page code This field identifies the parameter page as a disconnect-reconnect page. Set it to $02.

Page length This field contains the length, in bytes, of the data portion of the parameter page, not including bytes 0–5. Set it to $0A.

Figure 7-14 The MODE SELECT disconnect-reconnect parameter page



Buffer full ratio

Indicates to the scanner how full the scanner's image buffer must be before the scanner returns image data to the host computer in response to a GET DATA STATUS command (with the Wait bit set to 1). This field provides the numerator of a fraction whose denominator is 255. The resulting ratio controls how full the scanner buffer is allowed to get before data is returned to your program. The recommended value of 128 sets this threshold at 50 percent.

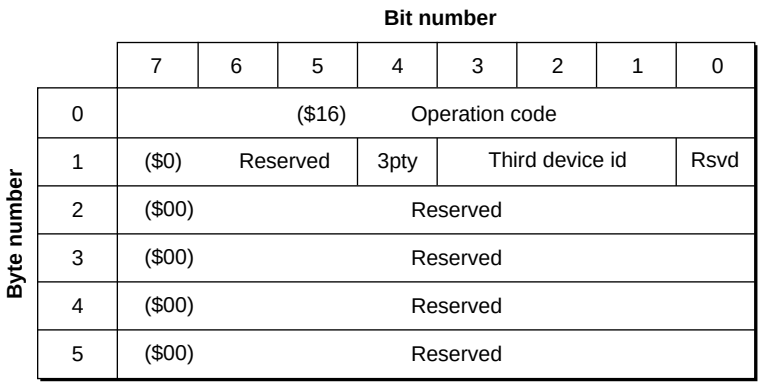
RESERVE ($16)

The RESERVE command prevents all devices other than the requesting device from accessing the scanner. The scanner rejects commands that it receives from devices other than the initiating host and instead returns a RESERVATION CONFLICT status. The host computer may reserve the target device for use by a third device by setting the third-party bit (3pty) in byte 1 of the command structure. In this case, only the third device can access the scanner to execute commands, although the initiating host computer may issue another RESERVE command or a RELEASE command. See "RELEASE ($17)," later in this chapter. Figure 7-15 shows the format of the RESERVE command structure.

The RESERVE command is terminated by the RELEASE command or by another RESERVE command issued from the original RESERVE command initiator.

The RESERVE and RELEASE commands together provide the basic mechanism for bus-contention resolution in a multidevice SCSI bus environment. The mechanism works as follows: an application running on a host computer must have uninterrupted access to the scanner while transferring commands and data to and from the scanner. The RESERVE command provides a means for an application to have access to the scanner until it is no longer needed. The application then issues a RELEASE command, which releases the scanner so that another SCSI device on the bus can use it.

Figure 7-15 The RESERVE command structure



FIELD DESCRIPTIONS

Here is a list of the fields and bits used in this command structure:

Operation code
: The operation code for the RESERVE command is $16.

Reserved
: These fields are reserved for future expansion. Set them to 0.

3pty This bit contains the third-device flag. When set to 1, this flag indicates that the scanner is to be reserved for use by a device other than the host computer. You specify the SCSI logical unit number for that other device in the Third device id field.

Third device id
This field specifies the SCSI logical unit number of the device for which the scanner is being reserved. Use this field only if you are reserving the scanner for another device and your program has set the 3pty bit to 1. Otherwise, set this field to 0.

RELEASE ($17)

The RELEASE command removes the specified reservation on the scanner. Only the device that issued the original RESERVE command can execute RELEASE. See "RESERVE ($16)," earlier in this chapter, for more information. Figure 7-16 shows the format of the RELEASE command structure.

Figure 7-16 The RELEASE command structure

Byte number	Bit number 7	6	5	4	3	2	1	0
0	($17) Operation code							
1	($0)	Reserved		3pty	Third device id			Rsvd
2	($00)	(0)	Reserved					
3	($00)	(0)	Reserved					
4	($00)	(0)	Reserved					
5	($00)	(0)	Reserved					

FIELD DESCRIPTIONS

Here is a list of the fields and bits used in this command structure:

Operation code
: The operation code for the RELEASE command is $17.

Reserved
: These fields are reserved for future expansion. Set them to 0.

3pty
: This bit contains the third-party device flag. When set to 1, this flag indicates that the scanner is to be released from use by a device other than the host computer. You specify the SCSI logical unit number for that other device in the Third device id field.

Third device id
: This field specifies the SCSI logical unit number of the device from which the scanner is being released. Use this field only if you are releasing the scanner for another device and your program has set the 3pty bit to 1. Otherwise, set this field to 0.

MODE SENSE ($1A)

The MODE SENSE command allows the host computer to read back the parameter information that was sent to the scanner with the MODE SELECT command. See "MODE SELECT ($15)," earlier in this chapter, for more information. Figure 7-17 shows the format of the MODE SENSE command structure. The scanner returns the information to the host computer in one or more MODE SENSE data pages as appropriate for the request and the type of scanner attached. The format of the data pages is described later in this section.

Figure 7-17 The MODE SENSE command structure

Byte number	Bit number 7	6	5	4	3	2	1	0
0	($1A) Operation code							
1	($0) Reserved			Page format	($0) Reserved			
2	($0) Reserved			Page code				
3	($00) Reserved							
4	Parameter list length							
5	($00) Reserved							

FIELD DESCRIPTIONS

Here is a list of the fields and bits used in this command structure:

Operation code
: The operation code for the MODE SENSE command is $1A.

Reserved
: These fields are reserved for future expansion. Set them to 0.

Page format
: This bit indicates that the return data is formatted into pages. Set it to 1.

Page code
: This field indicates which type of data page is to be returned.
Set it to 0 to request that the Apple-specific data page appropriate to the attached scanner be returned.
Set it to 2 to request that the disconnect-reconnect data page be returned.
Set it to $3F to request both pages.

Parameter list length
: This field contains the expected length, in bytes, of the data to be returned to the host computer. A transfer length of 0 indicates that no return data is to be passed. (This is not an error condition). The scanner stops transferring data when it has returned Parameter list length bytes or all appropriate MODE SENSE data, whichever comes first.

Apple-Specific Data Page Description for the Apple Scanner

Figure 7-18 shows the Apple-specific data page for the Apple Scanner.

Figure 7-18 The MODE SENSE Apple-specific data page for the Apple Scanner

Byte number	Bit number 7	6	5	4	3	2	1	0
0	MODE SENSE data length							
1	($00)			Reserved				
2	($00)			Reserved				
3	($00)			Reserved				
4	($00)			Page code				
5	($06)			Page length				
6	($0)			Reserved			Graymap	
7	Auto background-adjustment threshold							
8	($0)			Reserved				Lamp
9	($00)			Reserved				
10	($00)			Reserved				
11	($00)			Reserved				

Here is a list of the fields and bits used in this page:

MODE SENSE data length
: This field indicates the number of MODE SENSE data bytes that follow for this page. The value is $0B.

Reserved
: These fields are reserved for future expansion. They are set to 0.

Page code
: This field identifies the page type. It is set to $00.

Page length
: This field contains the length, in bytes, of the data portion of the parameter page, not including bytes 0–5. This field is set to 6.

Graymap
: This field contains the value that controls the graymap function of the scanner. The graymap function allows the scanner to enhance detail in light areas or dark areas of a document, or not to enhance either. A value of 0 results in more detail in the darker areas of the document. A value of 1 results in no alteration of the data. A value of 2 results in more detail in the lighter areas of the document.

Auto background-adjustment threshold

This field indicates the threshold level at which the scanner determines whether a dot is black or white when the user selects automatic background adjustment. The default threshold level is 64.

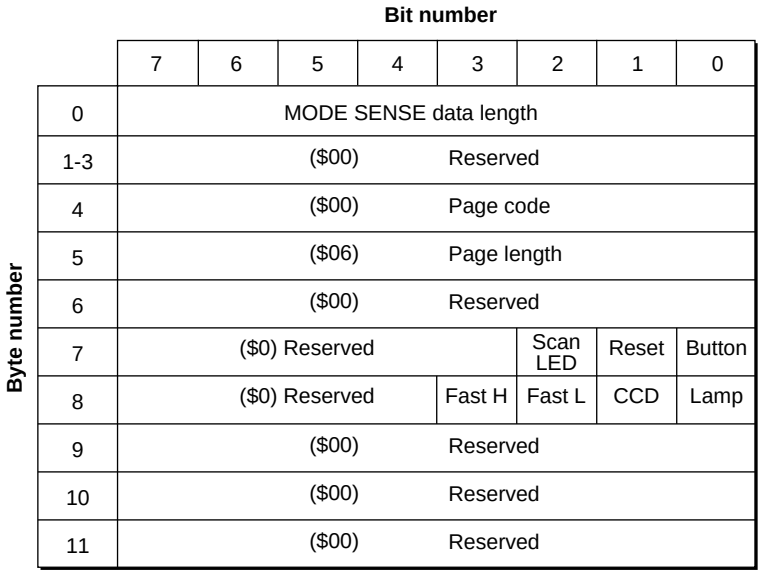
When excessively dark areas of an original document are scanned, automatic background adjustment constantly adjusts the brightness level, resulting in more detail in the darker areas. Light areas are left unaltered. To select automatic background adjustment, set the Threshold parameter to 0 in the DEFINE WINDOW PARAMETERS command parameter list. See "DEFINE WINDOW PARAMETERS ($24)" later in this chapter.

Lamp — This bit indicates the current state of the fluorescent lamp used to illuminate the document during scanning. A value of 1 indicates that the lamp is on; a value of 0 indicates that the lamp is off.

Apple-Specific Data Page Description for the OneScanner

Figure 7-19 shows the Apple-specific data page for the OneScanner.

Figure 7-19 The MODE SENSE Apple-specific data page for the OneScanner



Here is a list of the fields and bits used in this page:

MODE SENSE data length

This field indicates the number of MODE SENSE data bytes that follow for this page. The value is $0B.

Reserved — These fields are reserved for future expansion. Set them to 0.

Page code — This field identifies the page type. It is set to 0.

Page length This field contains the length, in bytes, of the data portion of the parameter page, not including bytes 0–5. This field is set to $06.

Scan LED This bit indicates the setting of the amber LED on the OneScanner. A value of 1 indicates that the LED is on. A value of 0 indicates that the LED is off.

Reset This bit indicates whether the scanner should reset the Button bit. It is used only for the MODE SELECT command. Do not use this bit. Applies only to scanners with ROM version 2.03, or earlier.

Button This bit indicates the current state of the button on the OneScanner since the last reset request (a MODE SELECT command with the Reset bit set to 1). A value of 1 indicates that the button has been pressed. A value of 0 indicates that the button has not been pressed. Applies only to scanners with ROM version 2.03, or earlier.

Fast H, Fast L

Together, these bits indicate the scan speed for the scanner. Here are four possible values for these bits:

Fast H	Fast L	Scan speed
0	0	Normal speed
0	1	High speed
1	0	Fast speed
1	1	Invalid value

At normal speed, the scanner processes the image at the slowest possible carriage speed and performs handshaking on all data exchanges with the host computer. At high speed, the scanner runs the carriage at high speed but still performs handshaking on all data exchanges with the host computer. At normal and high speeds, data handshaking prevents the loss of image data.

At fast speed, the scanner runs the carriage at high speed but does not perform any handshaking during data exchanges with the host. Eliminating the handshaking greatly improves the data exchange rate between the scanner and the host computer. However, image data may be lost if the host computer cannot keep up with the scanner. If the scanner loses data due to an overrun condition, it aborts the scan with a CHECK CONDITION status.

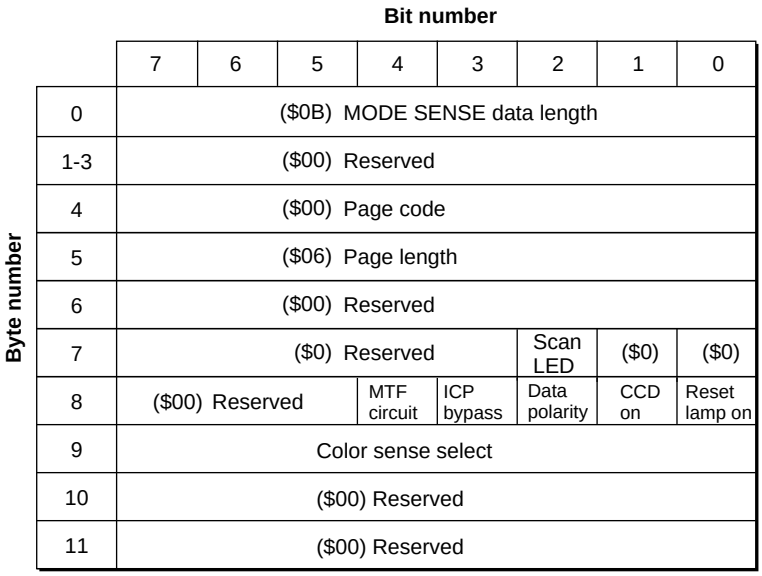
CCD This bit controls power to the CCD array in the scanner. A value of 1 causes the scanner to power on the CCD array whenever the scanner lamp is turned on. A value of 0 causes the scanner to power on the CCD array only in response to SCAN commands.

Lamp This bit indicates the current state of the fluorescent lamp used to illuminate the document during scanning. A value of 1 indicates that the lamp is on; a value of 0 indicates that the lamp is off.

Apple-Specific Data Page Description for the Color OneScanner

Figure 7-20 shows the Apple-specific data page for the Color OneScanner.

Figure 7-20 The MODE SENSE Apple-specific data page for the Color OneScanner



Here is a list of the fields and bits used in this page:

MODE SENSE data length
: This field indicates the number of MODE SENSE data bytes that follow for this page. The value is $0B.

Reserved
: These fields are reserved for future expansion. Set them to 0.

Page code
: This field identifies the page type. It is set to 0.

Page length
: This field contains the length, in bytes, of the data portion of the parameter page, not including bytes 0–5. Set it to $06.

Parameter length
: This field contains the expected length, in bytes, of the data to be returned to the host computer. A transfer length of 0 indicates that no return data is to be passed. This is not an error condition. The scanner stops transferring data when it has returned Parameter list length bytes, or all appropriate MODE SENSE data, whichever comes first.

Scan LED
: This bit indicates the setting of the amber LED which indicates the status of certain application functions. The default state for the LED is off.

MTF circuit
: This bit indicates the setting of the MTF circuit. If it is set to 1, the circuit is turned on. When it is reset to 0, the circuit is turned off.

ICP bypass This bit indicates the setting of the ICP. When it is set to 1, the circuit is bypassed. When it is reset to 0, the circuit is used.

Data polarity This bit indicates the polarity of the output image data. When it is set (1), 00 = white, and FF = black. When it is reset (0), 00 = black, and FF = white. The latter is the normal mode.

CCD on This bit indicates if the scanner has turned on power to the CCD (charge-coupled device) array when the scanning lamp is on. Whenever the scan command is issued, the CCD is powered up, even if the CCD on bit is set to 0.

Lamp on This bit, when set, indicates the scanner has turned on the lamp. The scanner goes through a shading correction before each scan, regardless of the state of the Lamp on bit. The lamp remains on until the next MODE SELECT signal is received, and the field is cleared. At this time, the lamp resumes normal on/off operation. If the lamp is on and there is no SCSI activity for more two minutes or more, the lamp turns off. When the scanner receives any SCSI command, the lamp turns on for a minimum of two minutes.

Color sense select

This field indicates which color sensor line or lines are selected for scanning. The green line sensor is the default for grayscale scanning.

SCAN ($1B)

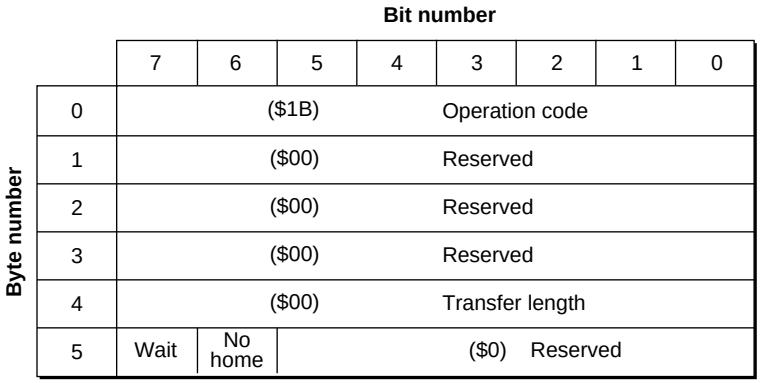
The SCAN command instructs the scanner to begin the scanning operation. Following the SCAN command structure, you may choose to send a Window identifier byte that identifies the scan area to process. See "DEFINE WINDOW PARAMETERS ($24)," later in this chapter, for more information.

For the Color OneScanner, there are two ways to abort a scan. The first way is to send the ABORT message. The second way is to issue the SCAN command, with the Transfer length parameter set to 0. For Apple Scanners and OneScanners, sending any command will abort the scan.

SCAN Command Structure for the Apple Scanner

Figure 7-21 shows the format of the SCAN command structure for the Apple Scanner.

Figure 7-21 The SCAN command structure for the Apple Scanner



FIELD DESCRIPTIONS

Here is a list of the fields and bits used in this command structure:

Operation code
: The operation code for the SCAN command is $1B.

Reserved
: These fields are reserved for future expansion. Set them to 0.

Transfer length
: This field indicates whether the application supplies a Window identifier byte after the 6 command bytes. A transfer length of 0 indicates that the application supplies no Window identifier byte and that no scan will take place. A transfer length of 1 indicates that the application supplies a Window identifier byte immediately after the command structure. The Window identifier byte must correspond to a Window identifier supplied to the DEFINE WINDOW PARAMETERS command.

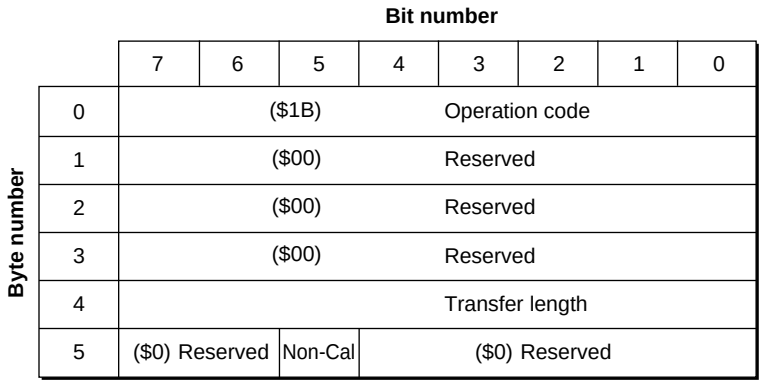
Wait If this bit is set to 1, the user must press the button on the scanner before the scanner will begin scanning.

No home If this bit is set to 1, the scanner lamp stays on and the carriage assembly remains where it stops at the end of a scan. After two minutes, if the scanner does not receive another SCAN command, the lamp goes off and the carriage assembly returns to the home position.

SCAN Command Structure for the OneScanner

Figure 7-22 shows the SCAN command structure for the OneScanner.

Figure 7-22 The SCAN command structure for the OneScanner



FIELD DESCRIPTIONS

Here is a list of the fields and bits used in this command structure:

Operation code
The operation code for the SCAN command is $1B.

Reserved These fields are reserved for future expansion. Set them to 0.

Transfer length
This field indicates whether the application supplies a Window identifier byte after the 6 command bytes. If the value of the transfer length is 0, the application supplies no Window identifier byte and no scan takes place. If the value of the transfer length is 1, the application supplies a Window identifier byte immediately after the command structure. The Window identifier byte must correspond to a Window identifier field value supplied to the DEFINE WINDOW PARAMETERS command.

Non-Cal Controls whether the scanner calibrates for the current lamp intensity before starting the scan. If this bit is set to 1, the scanner does not perform calibration. If this bit is set to 0, the scanner calibrates for the current lamp intensity before scanning. This calibration step takes a few seconds. Your application can reduce scan time by skipping this calibration step. However, skipping lamp calibration compromises scan quality. Consequently, your application does not use calibration except for preview scans.

SCAN Command Structure for the Color OneScanner

Figure 7-23 shows the format of the SCAN command structure for the Color OneScanner.

Figure 7-23 The SCAN command structure for the Color OneScanner

Table 1

Byte number	7	6	5	4	3	2	1	0
0	($1B) Operation code							
1	($00) Reserved							
2	($00) Reserved							
3	($00) Reserved							
4	Transfer length							
5	($00) Reserved							

FIELD DESCRIPTIONS

Here is a list of the fields used in this command structure:

Operation code
The operation code for the SCAN command is $1B.

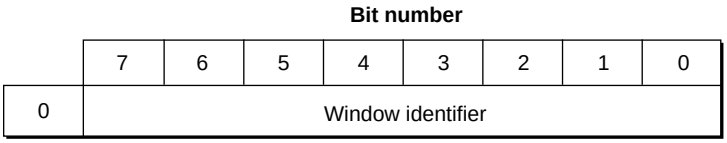
Reserved These fields are reserved for future expansion. Set them to 0.

Transfer length
This field indicates whether the application supplies a Window identifier byte after the 6 command bytes. A transfer length of 0 indicates that the application supplies no Window identifier byte and that no scan will take place. If a transfer length of 0 is sent to the scanner while a scanning operation is in progress, the scan operation will be aborted. A transfer length of 1 indicates that the application supplies a Window identifier byte immediately after the command structure. The Window identifier byte must correspond to a Window identifier supplied to the DEFINE WINDOW PARAMETERS command.

Window Parameter List

To cause the scanner to scan a document, you must append a Window identifier byte to the SCAN command structure. The Window identifier byte specifies the scan area and related scanning parameters, and it must correspond to a window identifier defined with the DEFINE WINDOW PARAMETERS command. Figure 7-24 shows the command byte format.

Figure 7-24 The SCAN command Window identifier byte



FIELD DESCRIPTION

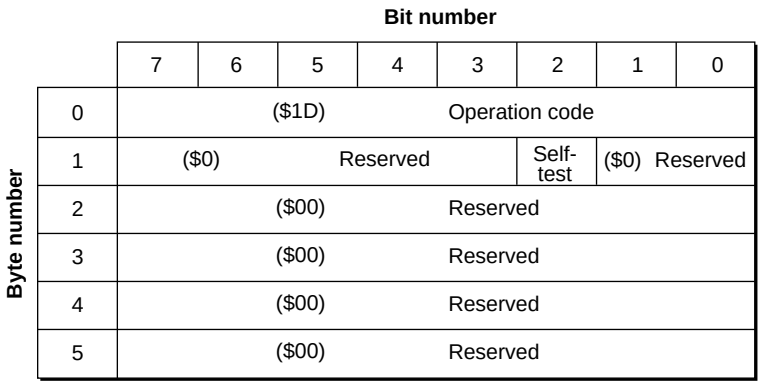
Window identifier

Specifies the scan area and related parameters for the scan operation. The value of the identifier byte must be the same as the value of the Window identifier byte indicated in the DEFINE WINDOW PARAMETERS command for that window.

SEND DIAGNOSTIC ($1D)

The SEND DIAGNOSTIC command directs the scanner to run a built-in self-test. Figure 7-25 shows the format of the SEND DIAGNOSTIC command structure.

Figure 7-25 The SEND DIAGNOSTIC command structure



FIELD DESCRIPTIONS

Here is a list of the fields and bits used in this command structure:

Operation code
: The operation code for the SEND DIAGNOSTIC command is $1D.

Reserved
: These fields are reserved for future expansion. Set them to 0.

Self-test
: Set this bit to 1 to request a self-test.

DEFINE WINDOW PARAMETERS ($24)

Before the scanner can perform a scan, the application program must provide certain details about the scan area. This information is provided in the form of parameters defining a window for each scan area. The DEFINE WINDOW PARAMETERS command passes this window-definition data to the scanner. The program defines the size, position, scanning resolution, scanning composition, and other parameters of each window. There are different parameter structures for each supported scanner. These structures are described later in this section.

IMPORTANT

The OneScanner and Color OneScanner do not support multiple windows. ▲

DEFINE WINDOW PARAMETERS Command Structure for the Apple Scanner

Figure 7-26 shows the format of the DEFINE WINDOW PARAMETERS command structure for the Apple Scanner.

Figure 7-26 The DEFINE WINDOW PARAMETERS command structure for the Apple Scanner

Byte number	Bit number 7	6	5	4	3	2	1	0
0	($24)				Operation code			
1 – 5	($00)				Reserved			
6	Transfer length (MSB)							
7	Transfer length							
8	Transfer length (LSB)							
9	Apple	($0)				Reserved		

FIELD DESCRIPTIONS

Here is a list of the fields and bits used in this command structure:

Operation code
: The operation code for the DEFINE WINDOW PARAMETERS command is $24.

Reserved
: These fields are reserved for future expansion. Set them to 0.

Transfer length

This field contains the length, in bytes, of the DEFINE WINDOW PARAMETERS parameter list. A transfer length of 0 indicates that the application program does not pass any parameters (this is not an error condition). The transfer length indicates the total length of all window descriptors and window parameter lists. The total transfer length is

$8 + (40 * n)$ bytes

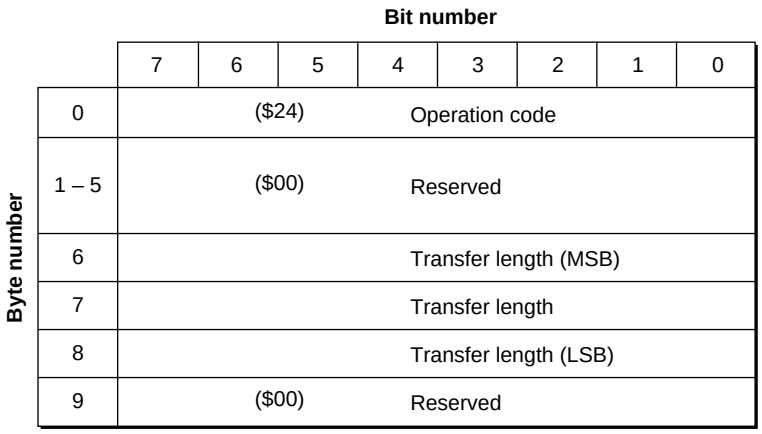
where n is the number of windows. Byte 6 of the structure contains the most-significant byte of the transfer length; byte 8 contains the least-significant byte of the transfer length.

Apple This bit, if set to 1, allows the Apple Scanner to recognize multiple window descriptors.

DEFINE WINDOW PARAMETERS Command Structure for the OneScanner and the Color OneScanner

Figure 7-27 shows the format of the DEFINE WINDOW PARAMETERS command structure for the OneScanner.

Figure 7-27 The DEFINE WINDOW PARAMETERS command structure for the OneScanner and the Color OneScanner



FIELD DESCRIPTIONS

Here is a list of the fields used in this command structure:

Operation code

The operation code for the DEFINE WINDOW PARAMETERS command is $24.

Reserved These fields are reserved for future expansion. Set them to 0.

Transfer length

This field contains the length, in bytes, of the DEFINE WINDOW PARAMETERS parameter list. If the value of the transfer length is 0, the application does not pass any parameters (this is not an error condition). The transfer length indicates the total length of all window descriptors and window parameter lists. The total transfer length must be set to $48 for the Apple Scanner and the OneScanner, and to $50 for the Color OneScanner. Byte 6 of the structure contains the most-significant byte of the transfer length; byte 8 contains the least-significant byte of the transfer length.

DEFINE WINDOW PARAMETERS Parameter List for the Apple Scanner

The window parameter list for the DEFINE WINDOW PARAMETERS command consists of one parameter list header and one or more unique window descriptors. When the user wants to scan more than one window, the application program must supply one window descriptor for each window. Figure 7-28 shows the format of the parameter list header and descriptor.

FIELD DESCRIPTIONS

The parameter list header describes the length, in bytes, of a window descriptor. An application program may transfer a maximum of 99 window descriptors. Here are the parameter list fields used in this structure:

Reserved These fields are reserved for future expansion. Set them to 0.

Parameter list length

This field specifies the length in bytes of a single window descriptor. Always set this field to 40 ($28).

The window descriptor contains information about one window. Here are the window descriptors used in this structure:

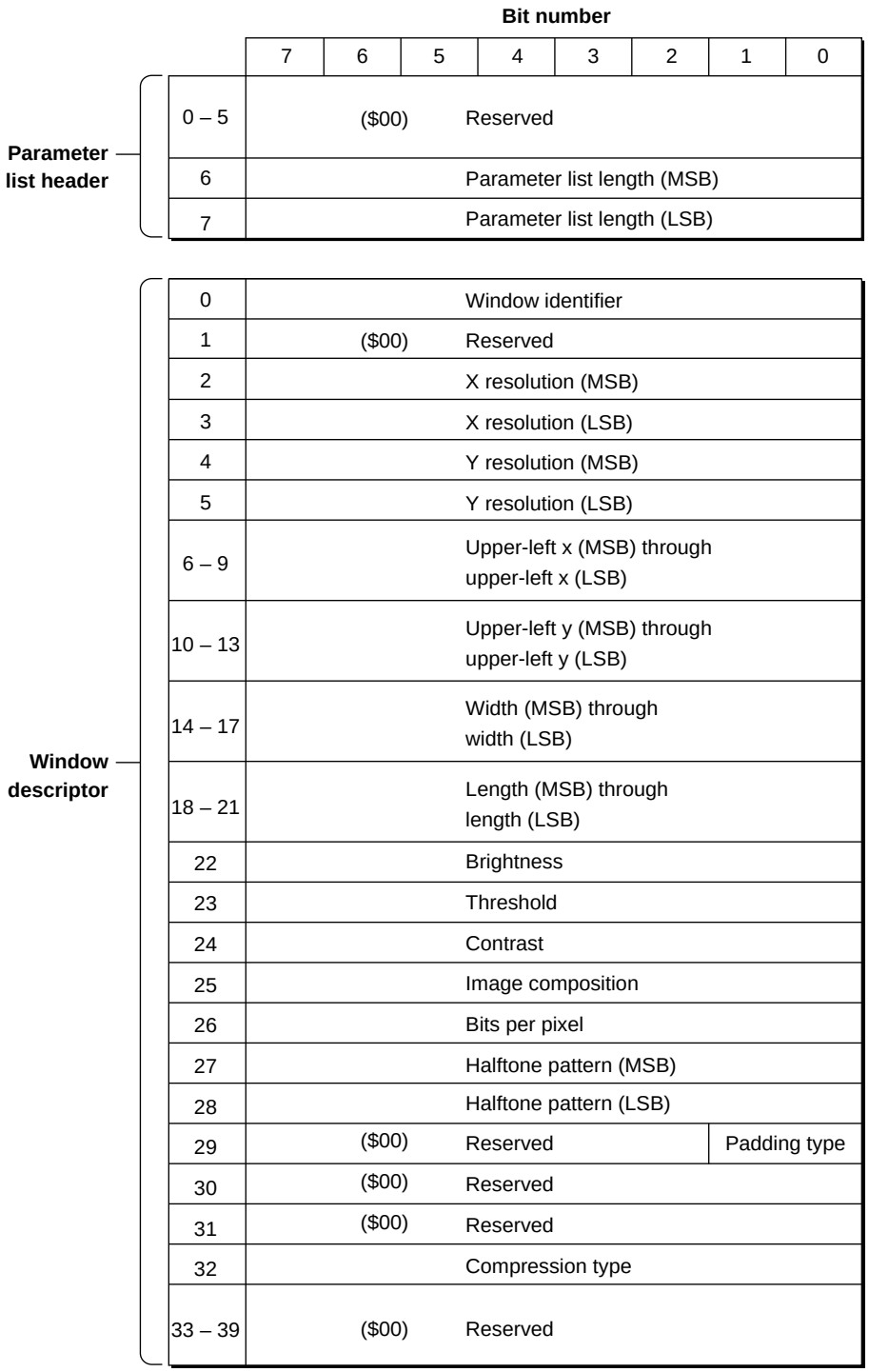
Window identifier

This field contains a number between 0 and 255, which uniquely identifies the window defined by the descriptor. Use this unique identifier to identify each window during data transfers and status requests.

Reserved These fields are reserved for future expansion. Set them to 0.

X resolution This field specifies the resolution, in dots per inch, along the horizontal direction (x-axis). In Line Art mode and Halftone mode, the Apple Scanner supports resolutions of 300, 285, 270, 255, 240, 225, 210, 200, 195, 180, 165, 150, 135, 120, 105, 100, 90, and 75 dpi. In Grayscale mode, the Apple Scanner supports resolutions of 300, 200, 150, 100, and 75 dpi. A value of 0 in this field indicates that the scanner uses a default horizontal resolution value of 75 dpi.

Figure 7-28 The DEFINE WINDOW PARAMETERS parameter list for the Apple Scanner



Y resolution	This field specifies the resolution, in dots per inch, in the vertical direction (y-axis). In Line Art mode and Halftone mode, the Apple Scanner supports resolutions of 300, 285, 270, 255, 240, 225, 210, 200, 195, 180, 165, 150, 135, 120, 100, 90, and 75 dpi. In Grayscale mode, the Apple Scanner supports resolutions of 300, 200, 150, 100, and 75 dpi. A value of 0 in this field indicates that the scanner uses a default vertical resolution value of 75 dpi.
Upper-left x	This field specifies the x-coordinate of the upper left corner of the rectangular window to be scanned. The point (0,0) is considered the upper-left corner of the window. Coordinates are measured from this point in increments of 1/1200 of an inch from the right edge of the scanner's glass surface (as viewed when you face the glass). Coordinates must be multiples of 8 times the x resolution; they must lie on byte boundaries. The default value of this field is 0.
Upper-left y	This field specifies the y-coordinate of the upper-left corner of the rectangular window to be scanned. The point (0,0) is considered the upper-left corner of the window. Coordinates are measured from this point in increments of 1/1200 of an inch from the top edge of the scanner's glass surface. The default value of this field is 0.
Width	This field specifies the window width in increments of 1/1200 of an inch along the horizontal axis. This value must be a multiple of 8 times the x resolution; the window border must lie on a byte boundary. The default value of this field is 10,208.
Length	This field specifies the window length in increments of 1/1200 of an inch along the vertical axis. The default value of this field is 13,200.
Brightness	This field specifies the brightness. A value of 0 results in the default value of 128. Any other value indicates a relative brightness: 255 is the highest setting, 1 is the lowest setting, and 128 is the average setting.
Threshold	This field specifies the threshold level. A value of 0 in Line Art mode causes the scanner to use automatic background adjustment. See "MODE SELECT ($15)," earlier in this chapter, for a description of the Auto background-adjustment threshold field. A value of 0 received in any other mode results in the use of the default setting. Any other value indicates a relative threshold parameter: 255 is the highest setting, 1 is the lowest setting, and 128 is the average setting. The default value for this parameter is 128.
Contrast	This field specifies the contrast at which the scanner scans the document. A value of 0 indicates that the scanner uses the default value of 1. Any other value indicates a relative contrast: 255 is the highest setting, 1 is the lowest setting, and 128 is the average setting. The contrast setting is valid only in Halftone mode and Grayscale mode.

Image composition

This field specifies the type of image data acquired. The default composition mode is Line Art mode. The following values are valid:

$00 Line Art mode

$01 Halftone mode

$02 Grayscale mode

Bits per pixel This field specifies, for any one pixel, the number of bits used to specify the visible density of the document being scanned. This field is valid only if the Image composition field indicates that gray-scale data is desired. The Apple Scanner supports 16 levels of gray in Grayscale mode and only black and white in Line Art mode and Halftone mode. Therefore, 4 bits per pixel are required to represent the full range of grays, and only 1 bit per pixel is required to represent either black or white.

Halftone pattern

This field specifies the halftone process by which the scanner converts multilevel gray tones to black and white dots. This parameter is only valid when the Image composition field specifies that halftone image data is desired. The following values are valid:

$00 Spiral

$01 Bayer

$02 Downloaded pattern

Padding type This field specifies what the scanner does if the image data transmitted to the host is not a whole number of bytes. This field must be $03, which truncates the data to the next byte boundary.

Compression type

This field specifies the compression technique that the scanner applies to the image data prior to transmission to the host computer. The default is no compression. The following values are valid:

$00 No compression

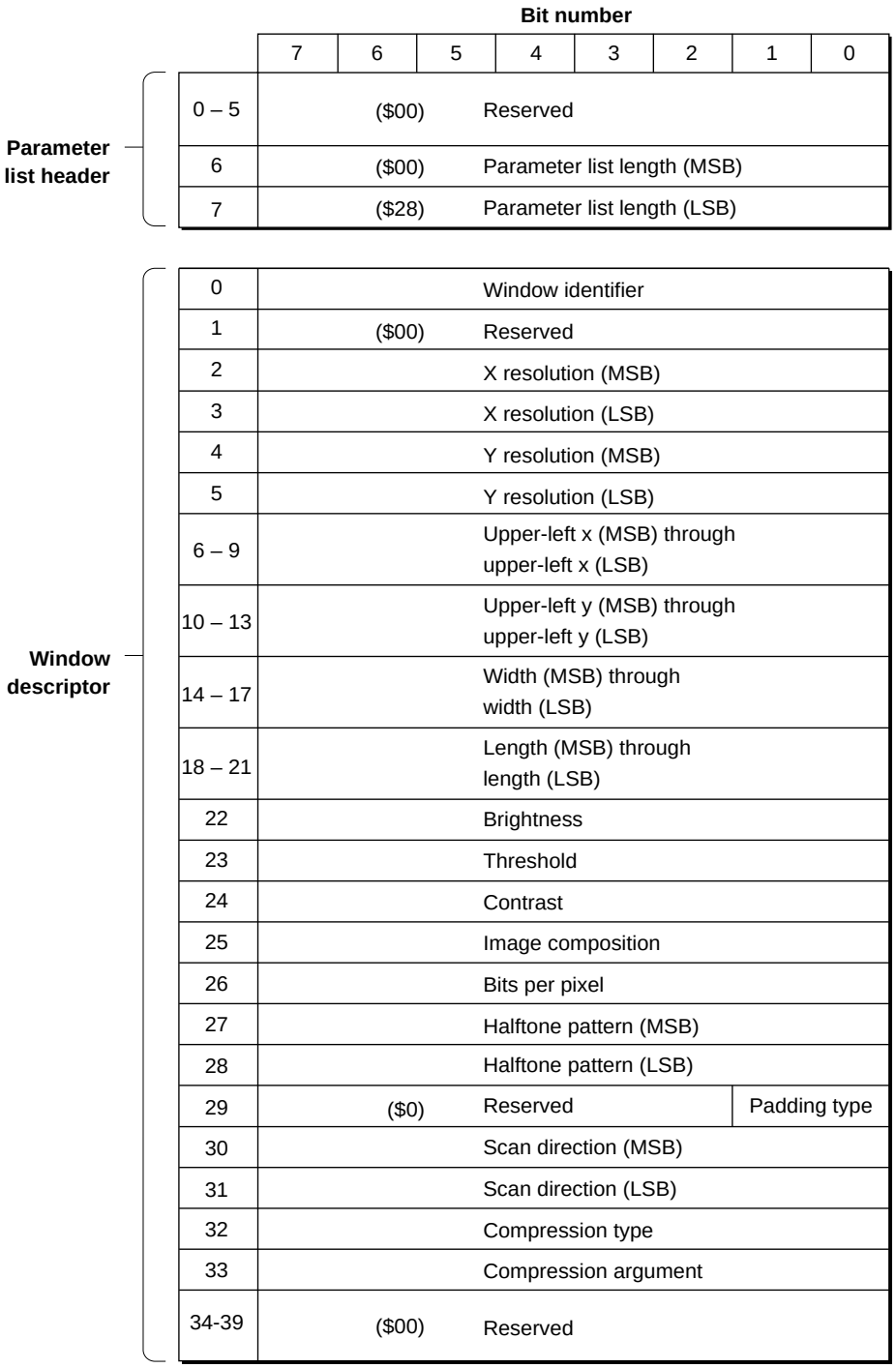
$01 CCITT Group III, one-dimensional

$83 White line skipped

DEFINE WINDOW PARAMETERS Parameter List for the OneScanner

The window parameter list for the DEFINE WINDOW PARAMETERS command consists of one parameter list header followed by at most one window descriptor. Figure 7-29 on page 160 shows the format of the parameter list header and descriptor for the OneScanner.

Figure 7-29 The DEFINE WINDOW PARAMETERS parameter list for the OneScanner



FIELD DESCRIPTIONS

The parameter list header describes the length, in bytes, of a window descriptor. An application program may transfer a maximum of one window descriptor. Here are the parameter list fields used in this structure:

Reserved — These fields are reserved for future expansion. Set them to 0.

Parameter list length
: This field specifies the length, in bytes, of a single window descriptor. Always set this parameter to 40 ($28).

The window descriptor contains information about one window. Here are the window descriptors used in this structure:

Window identifier
: This field identifies the window defined by the descriptor. This field must be set to 0.

Reserved — These fields are reserved for future expansion. Set them to 0.

X resolution — This field specifies the resolution, in dots per inch, along the horizontal direction (x-axis). The OneScanner supports horizontal resolutions that range from 72 to 300 dpi in 1-dpi increments. A value of 0 in this field indicates that the scanner uses a default horizontal resolution value of 72 dpi.

Y resolution — This field specifies the resolution, in dots per inch, in the vertical direction (y-axis). The OneScanner supports vertical resolutions that range from 72 to 300 dpi in 1-dpi increments. A value of 0 in this field indicates that the scanner uses a default vertical resolution value of 72 dpi.

Upper-left x — This field specifies the x-coordinate of the upper left corner of this rectangular window to be scanned. The point (0,0) is considered the upper left corner of the window. Coordinates are measured from this point in increments of 1/1200 of an inch from the right edge of the scanner's glass surface (as viewed when you face the glass). Coordinates must be multiples of 8 times the x resolution; they must lie on byte boundaries. The default value of this parameter is 0.

Upper-left y — This field specifies the y-coordinate of the upper left corner of this rectangular window to be scanned. The point (0,0) is considered the upper left corner of the window. Coordinates are measured from this point in increments of 1/1200 of an inch from the top edge of the scanner's glass surface. The default value of this parameter is 0.

Width — This field specifies the window width in increments of 1/1200 of an inch along the horizontal axis. This value must be a multiple of 8 times the x resolution; the window border must lie on a byte boundary. The default value of this parameter is 10,200.

Length — This field specifies the window length in increments of 1/1200 of an inch along the vertical axis. The default value of this parameter is 13,200.

Brightness This field specifies the brightness. A value of 0 results in the default value of 128. Any other value indicates a relative brightness: 255 is the highest setting, 1 is the lowest setting, and 128 is the average setting.

Threshold This field specifies the threshold level. A value of 0 results in use of the default setting. Any other value indicates a relative threshold parameter: 255 is the highest setting, 1 is the lowest setting, and 128 is the average setting. The default value for this parameter is 128.

Contrast This field specifies the contrast at which the scanner scans the document. A value of 0 indicates that the scanner uses the default value of 128. Any other value indicates a relative contrast: 255 is the highest setting, 1 is the lowest setting, and 128 is the average setting. The contrast setting is valid only in Halftone mode and Grayscale mode.

Image composition

This field specifies the type of image data acquired. The default composition mode is Line Art mode. The following values are valid:

$00	Line Art mode
$01	Halftone mode
$02	Grayscale mode

Bits per pixel This field specifies, for any one pixel, the number of bits used to specify the visible density of the document being scanned. This field is valid only if the Image composition field indicates that gray-scale data is desired. The OneScanner supports either 16 or 256 levels of gray in Grayscale mode. For 16-level gray support, set this field to 4; for 256-level gray support, set this field to 8.

Halftone pattern

This field specifies the halftone process by which the scanner converts multilevel gray tones to black and white dots. This parameter is only valid when the Image composition field specifies that halftone image data is desired. The following values are valid:

$00	4-by-4 Spiral
$01	4-by-4 Bayer
$02	Downloaded pattern
$03	8-by-8 Spiral
$04	8-by-8 Bayer

Padding type This field specifies what the scanner does if the image data transmitted to the host is not a whole number of bytes. This parameter must be $03, which truncates the data to the next byte boundary.

Scan direction This field specifies the direction for the scan operation. This field must be set to 0, which specifies that the scan is to proceed from left to right and top to bottom.

Compression type
: This field specifies the compression technique that the scanner applies to the image data prior to transmission to the host computer. This field must be set to 0, which indicates that the scanner is not to compress the image data.

Compression argument
: This field is reserved.

DEFINE WINDOW PARAMETERS Parameter List for the Color OneScanner

The window parameter list for the DEFINE WINDOW PARAMETERS command consists of one parameter list header followed by at most one window descriptor. Figure 7-30 on page 164 shows the format of the parameter list header and descriptor for the Color OneScanner.

FIELD DESCRIPTIONS

The parameter list header describes the length, in bytes, of a window descriptor. An application program may transfer a maximum of one window descriptor. Here are the parameter list fields used in this structure:

Reserved
: These fields are reserved for future expansion. Set them to 0.

Parameter list length
: This field specifies the length in bytes of a single window descriptor. Always set this parameter to 42($2A).

The window descriptor contains information about one window. Here are the window descriptors used in this structure:

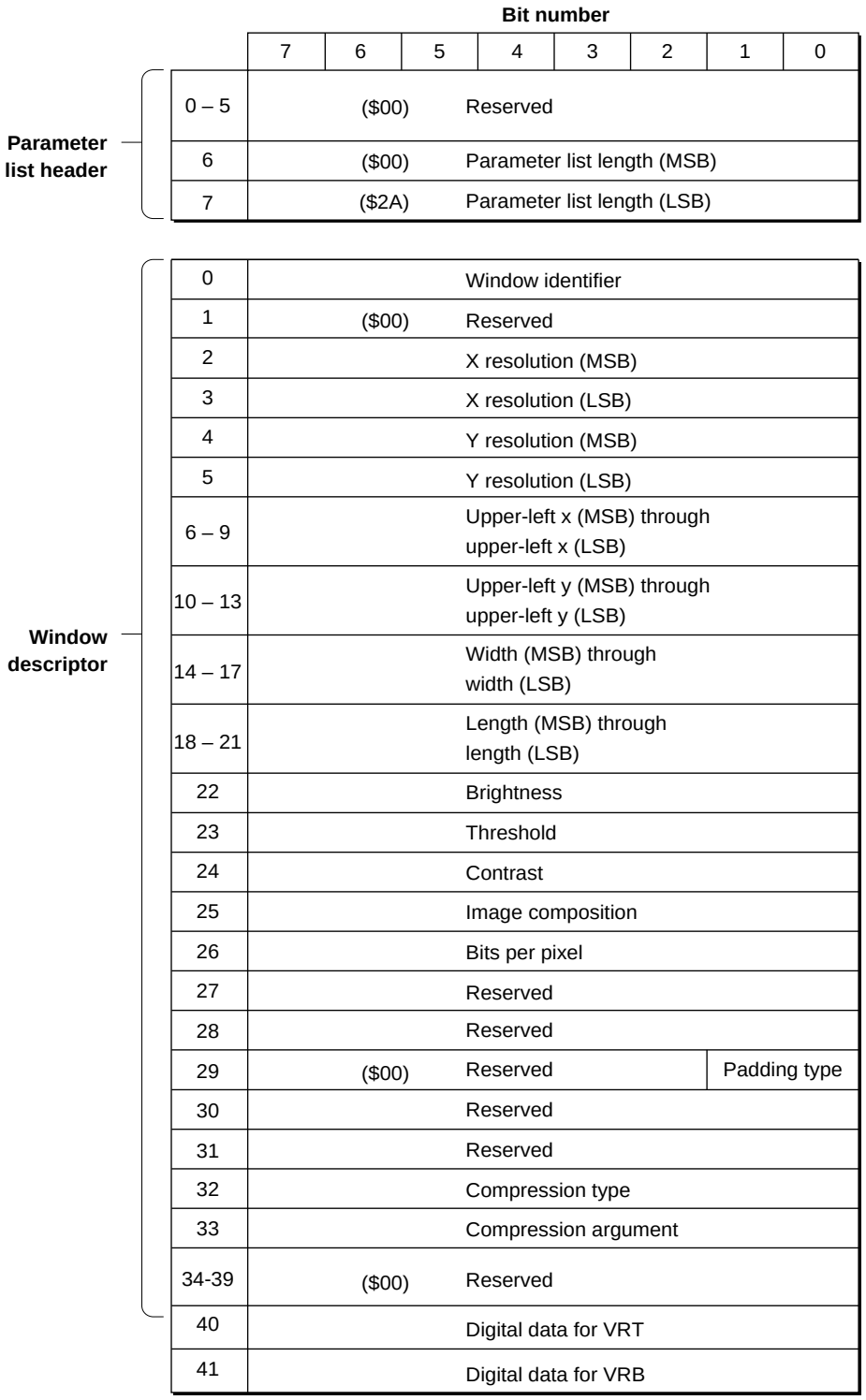
Window identifier
: This field identifies the window defined by the descriptor. This field must be set to 0.

Reserved
: These fields are reserved for future expansion. Set them to 0.

X resolution
: This field specifies the resolution, in dots per inch, along the horizontal direction (x-axis). The Color OneScanner supports horizontal resolutions that range from 72 to 300 dpi in increments of 1 dpi. A value of 0 in this field indicates that the scanner uses a default horizontal resolution value of 72 dpi.

Y resolution
: This field specifies the resolution, in dots per inch, in the vertical direction (y-axis). The Color OneScanner supports vertical resolutions that range from 72 to 300 dpi in 1-dpi increments. A value of 0 in this field indicates that the scanner uses a default vertical resolution value of 72 dpi.

Figure 7-30 The DEFINE WINDOW PARAMETERS parameter list for the Color OneScanner



	Bit number							
	7	6	5	4	3	2	1	0
Parameter list header								
0 – 5	($00) Reserved							
6	($00) Parameter list length (MSB)							
7	($2A) Parameter list length (LSB)							
Window descriptor								
0	Window identifier							
1	($00) Reserved							
2	X resolution (MSB)							
3	X resolution (LSB)							
4	Y resolution (MSB)							
5	Y resolution (LSB)							
6 – 9	Upper-left x (MSB) through upper-left x (LSB)							
10 – 13	Upper-left y (MSB) through upper-left y (LSB)							
14 – 17	Width (MSB) through width (LSB)							
18 – 21	Length (MSB) through length (LSB)							
22	Brightness							
23	Threshold							
24	Contrast							
25	Image composition							
26	Bits per pixel							
27	Reserved							
28	Reserved							
29	($00)			Reserved				Padding type
30	Reserved							
31	Reserved							
32	Compression type							
33	Compression argument							
34-39	($00) Reserved							
40	Digital data for VRT							
41	Digital data for VRB							

Upper-left x This field specifies the x-coordinate of the upper left corner of the rectangular window to be scanned. The point (0,0) is considered the upper left corner of the window. Coordinates are measured from this point in increments of 1/1200 of an inch from the right edge of the scanner's glass surface (as viewed when you face the glass). Coordinates must be multiples of 8 times the x resolution; they must lie on byte boundaries. The default value of this parameter is 0.

Upper-left y This field specifies the y-coordinate of the upper left corner of the rectangular window to be scanned. The point (0,0) is considered the upper left corner of the window. Coordinates are measured from this point in increments of 1/1200 of an inch from the top edge of the scanner's glass surface. The default value of this parameter is 0.

Width This field specifies the window width in increments of 1/1200 of an inch along the horizontal axis. This value must be a multiple of 8 times the x resolution; the window border must lie on a byte boundary. The default value of this parameter is 10,200.

Length This field specifies the window length in increments of 1/1200 of an inch along the vertical axis. The default value of this parameter is 13,200.

Brightness This field specifies the brightness by specifying the offset voltage or value of the analog-to-digital converter. A value of 0 results in the default value of 128. Any other value indicates a relative brightness: 255 is the lowest voltage setting, giving the highest brightness; 1 is the highest voltage setting, giving the lowest brightness; 128 is the average setting.

Threshold This field specifies the threshold level. A value of 0 results in use of the default setting. Any other value indicates a relative threshold parameter: 255 is the highest setting, 1 is the lowest setting, and 128 is the average setting. The default value for this parameter is 128.

Contrast This field specifies the contrast at which the scanner scans the document. A value of 0 indicates that the scanner uses the default value of 128. Any other value indicates a relative contrast: 255 is the highest setting, 1 is the lowest setting, and 128 is the average setting. The contrast setting is valid only in Grayscale mode.

Image composition

This field specifies the type of image data acquired. The default is Line Art mode. Here are the valid values:

$00	Line Art mode (bi-level black and white)
$02	Grayscale mode (multi-level black and white)
$03	Bi-level Color mode (RGB)
$05	Full Color mode (RGB)

Bits per pixel This field specifies, for any one pixel, the number of bits used to specify the visible density of the document being scanned. This field is only valid if the Image composition field indicates grayscale data is desired. The Color OneScanner supports 4 ($04) and 8 ($08) bits of gray-scale or color data, 3 bits for bi-level color, and 24 bits for full color (8 bits per color). For 16-level gray support, set this field to 4; for 256-level gray support, set this field to 8.

Padding type This field specifies what the scanner does if the image data transmitted to the host is not a whole number of bytes. This parameter must be $03, which truncates the data to the next byte boundary.

Compression type
This field specifies the compression technique that the scanner applies to the image data prior to transmission to the host computer. This field must be set to 0, which indicates that the scanner is not to compress the image data.

Compression argument
This field is reserved.

Digital data for Vrt and Vrb
These fields determine the contrast range of a scanned image. Vrt (voltage reference top) determines the upper end of the contrast range, and Vrb (voltage reference bottom) the lower end of the contrast range. If there is a value of 0 in either Vrt or Vrb, Vrt and Vrb are disabled, and Brightness and Contrast are used instead. Any other setting between 1 and 255 indicates the scanner should use the related value, for example, if the setting is 3, the scanner should use the value 3, and so forth.

Using Window Descriptors With the Apple Scanner

Note
This section applies only to the Apple Scanner. ◆

If the Apple bit in a DEFINE WINDOW PARAMETERS command structure is set to 1, an application program may send more than one window descriptor. The first window descriptor defines the primary window as well as the parameters for the entire scan. The additional window descriptors define secondary windows within the primary window, each of which may have a different image composition type. For example, the primary window could be used to define a scan as Line Art mode, and you may also define secondary windows within the primary window as Halftone composition.

Since halftone data is basically just a black-and-white image, the halftone data and line art data can be mixed freely. However, only halftone data and line art data can be mixed. Mixing with grayscale data is not allowed. If windows overlap, the window descriptor with the higher window identifier value takes priority.

Additional window descriptors must have all fields set to the same value as those in the primary window descriptor, except for the five fields that define the window rectangle (upper-left y, upper-left x, Width, Length, and Window identifier) and the Composition field. Window descriptor fields with different values result in an error. The byte-boundary restriction that applies to the parameters Width and upper-left x of the primary window does not apply to those parameters for secondary windows.

You may specify any window identifier as a parameter in the DEFINE WINDOW PARAMETERS command. All other commands—such as SCAN, GET DATA STATUS, and READ—accept only the window identifier of a primary window.

GET WINDOW PARAMETERS ($25)

The GET WINDOWS PARAMETERS command provides a means for the initiator to get information about a window previously defined using the DEFINE WINDOW PARAMETERS command. Figure 7-31 shows the format of the GET WINDOW PARAMETERS command structure.

Note
This command is supported only by the Color OneScanner. ◆

Figure 7-31 The GET WINDOW PARAMETERS command structure

Byte number	Bit number 7	6	5	4	3	2	1	0
0	($25)			Operation code				
1	($00)			Reserved				
2	($00)			Reserved				
3	($00)			Reserved				
4	($00)			Reserved				
5	($00)			Reserved				
6	Transfer length (MSB)							
7	Transfer length							
8	Transfer length (LSB)							
9	($00)			Reserved				

FIELD DESCRIPTIONS

Here is a list of the fields and bits used in this command structure:

Operation code
: The operation code for the GET WINDOW PARAMETERS command is $25.

Reserved
: These fields are reserved for future expansion. Set them to 0.

Transfer length
: These fields specify the length, in bytes, of the window description information sent to the initiator. If the value of the transfer length is 0, no window description information was sent. This is not considered an error. The scanner stops transferring data when it has returned data equivalent to the transfer byte length, or when it has returned all appropriate GET WINDOWS data, whichever comes first.

Figure 7-32 shows the GET WINDOW PARAMETERS parameter list.

Figure 7-32 The GET WINDOW PARAMETERS parameter list

Byte	Bit 7	6	5	4	3	2	1	0
0	Data transfer length (MSB)							
1	Data transfer length (LSB)							
2	($00)			Reserved				
3	($00)			Reserved				
4	($00)			Reserved				
5	($00)			Reserved				
6	($00)			Window descriptor length (MSB)				
7	($2A)			Window descriptor length (LSB)				
				Window descriptor(s)				
0				Window identifier				
1	($00)			Reserved				
2				X resolution (MSB)				
3				X resolution (LSB)				
4				Y resolution (MSB)				
5				Y resolution (LSB)				
6				Upper left X (MSB)				
7				Upper left X				
8				Upper left X				
9				Upper left X (LSB)				
A				Upper left Y (MSB)				
B				Upper left Y				
C				Upper left Y				
D				Upper left Y (LSB)				
E				Width (MSB)				
F				Width				
10				Width				
11				Width (LSB)				
12				Length (MSB)				
13				Length				
14				Length				
15				Length (LSB)				
16				Brightness				
17				Threshold				
18				Contrast				
19				Image composition				
1A				Bits per pixel				
1B	($00)			Reserved				
1C	($00)			Reserved				
1D	($00)			Reserved		(3)		Padding type
1E	($00)			Reserved				
1F	($00)			Reserved				
20	($00)			Compression type				
21	($00)			Compression argument				
22	($00)			Reserved				
23	($00)			Reserved				
24	($00)			Reserved				
25	($00)			Reserved				
26	($00)			Reserved				
27	($00)			Reserved				
28				DD (Digital data) for VRT				
29				DD (Digital data) for VRB				

READ ($28)

The READ command instructs the scanner to send data currently in its memory to the host computer. With the Apple Scanner, the READ command may return only image data. With the OneScanner, the READ command may return image data or the contents of the scanner's calibration RAM or static RAM. The Color OneScanner may return image data, the 3-by-3 matrix for color correction, the gamma function table, and PSRAM data.

Use the GET DATA STATUS command to determine the amount of image data available for a particular window. See "GET DATA STATUS ($34)," later in this chapter, for more information.

READ Command Structure for the Apple Scanner

Figure 7-33 shows the format of the READ command structure for the Apple Scanner.

Figure 7-33 The READ command structure for the Apple Scanner

Byte number	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	($28)				Operation code			
1	($00)				Reserved			
2	($00)				Transfer data type			
3	($00)				Reserved			
4	Transfer ID (MSB)							
5	Transfer ID (LSB)							
6	Allocation length (MSB)							
7	Allocation length							
8	Allocation length (LSB)							
9	($00)				Reserved			

FIELD DESCRIPTIONS

Here is a list of the fields used in this command structure:

Operation code
: The operation code for the READ command is $28.

Reserved
: These fields are reserved for future expansion. Set them to 0.

Transfer data type

This field specifies the type of data to be read. It must be set to $00.

Transfer ID This field specifies the window for an image data read operation. Before the scanner can read data from a window, your application program must define the window in a DEFINE WINDOW PARAMETERS command and then include it in the window list of a SCAN command.

Allocation length

This field specifies the maximum number of bytes that the host computer has allocated for the returned data. If the allocation length is 0, the scanner transfers no data (this is not an error condition). The command is terminated either when the host computer receives Allocation length bytes or when the scanner sends all available data, whichever is less. The GET DATA STATUS command determines the amount of data available for a given window.

READ Command Structure for the OneScanner

Figure 7-34 shows the READ command structure for the OneScanner. When reading the OneScanner calibration RAM, you must issue a read request for 2,550 bytes. When reading a downloaded halftone pattern from the OneScanner static RAM, you may read all or part of the pattern.

Figure 7-34 The READ command structure for the OneScanner

Byte number	Bit number 7	6	5	4	3	2	1	0
0	($28) Operation code							
1	($00) Reserved							
2	($00 or $02 or $80) Transfer data type							
3	($00) Reserved							
4	($00) Transfer ID (byte 1)							
5	($00) Transfer ID (byte 2)							
6	Allocation length (MSB)							
7	Allocation length							
8	Allocation length (LSB)							
9	($00) Reserved							

FIELD DESCRIPTIONS

Here is a list of the fields used in this command structure:

Operation code

The operation code for the READ command is $28.

Reserved — These fields are reserved for future expansion. Set them to 0.

Transfer data type

This field specifies the type of data to be read. Set this field to one of the following values:

$00	Read image data
$02	Read halftone pattern from static RAM
$80	Read calibration RAM

Transfer ID — This field specifies the window for an image data read operation. Before the scanner can read data from a window, your application program must define the window in a DEFINE WINDOW PARAMETERS command and then include it in the window list of a SCAN command.

Allocation length

This field specifies the maximum number of bytes that the host computer has allocated for the returned data. If the allocation length is 0, the scanner transfers no data (this is not an error condition). The command is terminated either when the host computer receives Allocation length bytes or when the scanner sends all available data, whichever is less. The GET DATA STATUS command determines the amount of data available for a given window.

When reading the contents of the processor static RAM (Transfer data type set to $02), you may read some or all of a downloaded halftone pattern. Set Allocation length to a value less than or equal to the size of the stored halftone matrix plus the dimension byte (see "Halftone Parameter Page Description," later in this chapter, for a detailed description of the format of a halftone matrix). You may not read the contents of any of the built-in halftone patterns.

When reading the contents of calibration RAM (Transfer data type set to $80), you may read some or all of that memory. Set Allocation length to a value in the range from 1 through 255. Each returned byte contains the calibration data for the corresponding pixel sensor in the CCD array.

READ Command Structure for the Color OneScanner

Figure 7-35 shows the READ command structure for the Color OneScanner. When reading the Color OneScanner calibration RAM, you must issue a read request for 8,100 bytes. When reading the 3-by-3 matrix table, you must issue a read request for 18 bytes. When reading the gamma table, you must issue a read request for 768 bytes.

Figure 7-35 The READ command structure for the Color OneScanner

Byte number	Bit number 7	6	5	4	3	2	1	0	
0				($28)					Operation code
1				($00)					Reserved
2				($00 or $01 or $03 or $80)					Transfer data type
3				($00)					Reserved
4				($00)					Transfer ID (byte 1)
5				($00 or $02)					Transfer ID (byte 2)
6									Allocation length (MSB)
7									Allocation length
8									Allocation length (LSB)
9				($00)					Reserved

FIELD DESCRIPTIONS

Here is a list of the fields used in this command structure:

Operation code

The operation code for the READ command is $28.

Reserved These fields are reserved for future expansion. Set them to 0.

Transfer data type

This field specifies the type of data to be read. Set this field to one of the following values:

$00 Read image data

$01 Read 3-by-3 matrix for color correction

$03 Read gamma function table

$80 Read calibration RAM

Transfer ID This field specifies the window for an image data read operation. Before the scanner can read data from a window, your application program must define the window in a DEFINE WINDOW PARAMETERS command and then include it in the window list of a SCAN command. The first byte of the Transfer ID field is set to $00, meaning only one window is supported. The second byte of the Transfer ID field differentiates between data transfers of the same Transfer data type, as shown in Table 7-2.

Allocation length

This field specifies the maximum number of bytes that the host computer has allocated for the returned data. If the allocation length is 0, the scanner transfers no data (this is not an error condition). The command is terminated either when the host computer receives Allocation length bytes or when the scanner sends all available data, whichever is less. The GET DATA STATUS command determines the amount of data available for a given window.

When reading the contents of the calibration RAM (Transfer data type set to $80), you may read some or all of that memory. Set Allocation length to 8100. Each returned byte contains the calibration data for the corresponding pixel sensor in the CCD array. When reading the 3-by-3 matrix table, set the Allocation length to 18 bytes. When reading the Gamma table, set the Allocation length to 768 bytes

When reading image data, the data transferred must always be an even number of bytes. If the initiator asks for an odd number of bytes, the scanner returns a CHECK CONDITION status. If the initiator then sends a REQUEST SENSE command, the scanner returns SENSE KEY $5, indicating that this is an illegal transaction. If the initiator sends SCSI commands REQUEST SENSE, INQUIRY, or GET WINDOW PARAMETERS, it can then ask for either an odd or an even number of bytes of data.

Table 7-2 Transfer data types and transfer identification

Transfer data type	Transfer ID (and byte)	Description
$00	$00	Image data
$01	$02	3-by-3 color correction matrix
$03	$02	Gamma function table
$80	$02	PSRAM table for red/green/blue

The internal hardware structure of the scanner requires that each color of each line, or each line, in the case of monochrome modes, is composed of an even number of bytes. If the scan area requested results in a line that is not an even number of bytes in length, the scanner increases the scan width so that the scan line is an even number of bytes. Figure 7-36 illustrates what happens in this case. The host then adds the additional number of bytes, color, and lines to the originally requested scan width when reading the image data.

Figure 7-36 Extra bit or byte returned for different resolutions and composition modes

Number of pixels per line	Image composition monochrome	Additional bit or byte per line	Image composition color	Additional bit or byte			
				R	G	B	Total
16N	1 bit	0 bits	3 bit	0 bits	0 bits	0 bits	0 bits
16N+1		15 bits (15 pixels)		15 bits	15 bits	15 bits	45 bits (15 pixels)
16N+2		14 bits (14 pixels)		14 bits	14 bits	14 bits	42 bits (14 pixels)
16N+3		13 bits (13 pixels)		13 bits	13 bits	13 bits	39 bits (13 pixels)
16N+4		12 bits (12 pixels)		12 bits	12 bits	12 bits	36 bits (12 pixels)
16N+5		11 bits (11 pixels)		11 bits	11 bits	11 bits	33 bits (11 pixels)
16N+6		10 bits (10 pixels)		10 bits	10 bits	10 bits	30 bits (10 pixels)
16N+7		9 bits (9 pixels)		9 bits	9 bits	9 bits	27 bits (9 pixels)
16N+8		8 bits (8 pixels)		8 bits	8 bits	8 bits	24 bits (8 pixels)
16N+9		7 bits (7 pixels)		7 bits	7 bits	7 bits	21 bits (7 pixels)
16N+10		6 bits (6 pixels)		6 bits	6 bits	6 bits	18 bits (6 pixels)
16N+11		5 bits (5 pixels)		5 bits	5 bits	5 bits	15 bits (5 pixels)
16N+12		4 bits (4 pixels)		4 bits	4 bits	4 bits	12 bits (4 pixels)
16N+13		3 bits (3 pixels)		3 bits	3 bits	3 bits	9 bits (3 pixels)
16N+14		2 bits (2 pixels)		2 bits	2 bits	2 bits	6 bits (2 pixels)
16N+15		1 bits (1 pixels)		1 bits	1 bits	1 bits	3 bits (1 pixel)
4N	4 bits	0 bits					
4N+1		12 bits (3 pixels)					
4N+2		8 bits (2 pixels)					
4N+3		4 bits (1 pixels)					
8N	8 bits	0 bytes	24 bits	0 bytes	0 bytes	0 bytes	0 bytes
8N+1		1 byte (1 pixel)		1 byte	1 byte	1 byte	3 bytes (1 pixel)

SEND ($2A)

The SEND command provides a means for the scanner to receive parameter data from the host computer. With the Apple Scanner, you can use this command to download halftone matrixes. With the OneScanner, you can use this command to download halftone matrixes or to set the contents of the scanner's calibration RAM. The Color OneScanner downloads a 3-by-3 matrix for color correction, the gamma table, and PSRAM data write. This section describes the SEND command structures for the Apple Scanner, the Apple OneScanner, and the Apple Color OneScanner. It also details the halftone matrix format for the Scanner and the OneScanner, and the Color OneScanner's 3-by-3 matrix, gamma table, and PSRAM data write.

SEND Command Structure for the Apple Scanner

Figure 7-37 shows the format of the SEND command structure for the Apple Scanner.

Figure 7-37 The SEND command structure for the Apple Scanner

Byte number	Bit number 7	6	5	4	3	2	1	0
0	($2A)				Operation code			
1	($00)				Reserved			
2	($02)				Transfer data type			
3	($00)				Reserved			
4	($00)				Reserved			
5	($02)				Transfer ID byte			
6	($00)				Reserved			
7	($00)				Transfer length (MSB)			
8	($11)				Transfer length (LSB)			
9	($00)				Reserved			

FIELD DESCRIPTIONS

Here is a list of the fields used in this command structure:

Operation code
: The operation code for the SEND command is $2A.

Reserved
: These fields are reserved for future expansion. Set them to 0.

Transfer data type
: This field indicates the type of parameter data to be transferred to the scanner. Only halftone parameter data may be transferred to the Apple Scanner. Set this field to $02.

Transfer ID byte

This field indicates which halftone matrix within the scanner will be altered. This field must always be $02.

Transfer length

This field indicates the amount of parameter data to be sent. Since the Apple Scanner supports this command only for 4-by-4 halftone matrixes, this parameter must always be set to 17 ($11)—the size of a 4-by-4 halftone parameter page.

SEND Command Structure for the OneScanner

Figure 7-38 shows the SEND command structure for the OneScanner.

Figure 7-38 The SEND command structure for the OneScanner

Byte number	Bit number 7	6	5	4	3	2	1	0
0	($2A)				Operation code			
1	($00)				Reserved			
2	($02 or $80)				Transfer data type			
3	($00)				Reserved			
4	Transfer ID (MSB)							
5	Transfer ID (LSB)							
6	($00)				Reserved			
7	Transfer length (MSB)							
8	Transfer length (LSB)							
9	($00)				Reserved			

FIELD DESCRIPTIONS

Here is a list of the fields used in this command structure:

Operation code

Set the operation code to $2A.

Reserved These fields are reserved for future expansion. Set them to 0.

Transfer ID This field indicates which halftone matrix within the scanner will be altered. This field must always be $02.

Transfer data type

This field indicates the type of parameter data to be transferred to the scanner. Set this field to one of the following values:

$02 Send halftone pattern to static RAM

$80 Set calibration RAM

Transfer length

This field indicates the amount of parameter data to be sent. For halftone matrixes, set this field to the size of the matrix plus the dimension byte. (See "Halftone Parameter Page Description," later in this chapter, for a detailed description of the format of a halftone matrix.) The OneScanner supports both 4-by-4 and 8-by-8 halftone matrixes.

When setting the contents of the calibration RAM, you must set Transfer length to 2550. The parameter data consists of a contiguous buffer of 2550 bytes. Each byte of parameter data contains the new calibration data for the corresponding pixel sensor in the CCD array.

SEND Command Structure for the Color OneScanner

Figure 7-39 shows the SEND command structure for the Color OneScanner.

Figure 7-39 The SEND command structure for the Color OneScanner

Byte number	Bit number 7	6	5	4	3	2	1	0	
0	($2A)								Operation code
1	($00)								Reserved
2	($01 or $03 or $80)								Transfer data type
3	($00)								Reserved
4	($00)								Transfer ID (byte 1)
5	($02)								Transfer ID (byte 2)
6	($00)								Reserved
7									Transfer length (MSB)
8									Transfer length (LSB)
9	($00)								Reserved

FIELD DESCRIPTIONS

Here is a list of the fields used in this command structure:

Operation code

The operation code for the SEND command is $2A.

Reserved These fields are reserved for future expansion. Set them to 0.

Transfer data type

This field indicates the type of parameter data to be transferred to the scanner. Set this field to one of the following values:

$01 Send 3-by-3 matrix for color correction

$03 Send gamma table

$80 Set calibration RAM (PSRAM)

Transfer ID This field is always set to $02.

Transfer length

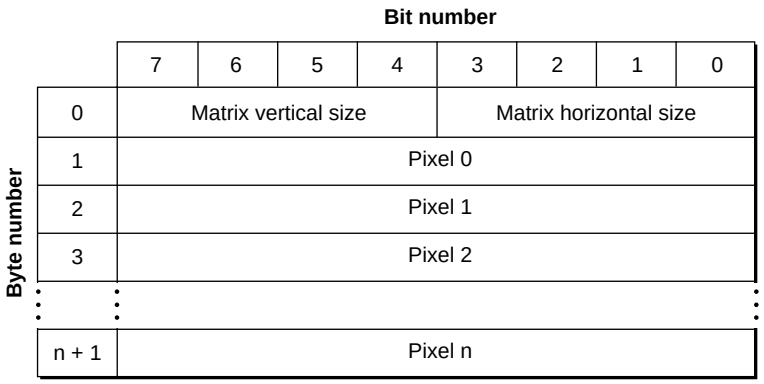
This field indicates the amount of parameter data to be sent.

If the transfer type is $80 (set calibration RAM), you must set the transfer length to 8100 bytes, to update the PSRAM calibration data for all three lines of sensors. Each single byte of data contains the new calibration data (CD) for each CD pixel sensor in ascending order. Set the transfer length to 18 for the 3-by-3 matrix, and to 768 for the gamma correction table.

Halftone Parameter Page Description for the Apple Scanner and the OneScanner

The halftone parameter page, shown in Figure 7-40, defines the halftone matrix that the application program downloads to the scanner. The size of the parameter page varies according to the dimensions of the halftone matrix. The parameter page contains a single byte that indicates the matrix dimensions, followed by 1 byte of data for each matrix element. Thus, for a 4-by-4 matrix, the parameter page contains 17 bytes of data (the dimension byte followed by 16 bytes of matrix data), the parameter page for an 8-by-8 matrix contains 65 bytes, and so on.

Figure 7-40 The SEND command halftone parameter page



Matrix vertical size, Matrix horizontal size

These fields indicate the dimensions of the matrix. The dimensions are encoded in a single byte, as follows: bits 4 through 7 contain the y dimension and bits 0 through 3 contain the x dimension. For a 4-by-4 matrix, set this field to $44. For an 8-by-8 matrix, set this field to $88. Figure 7-41 shows how the pixel numbers map onto a 4-by-4 matrix; pixels map similarly for an 8-by-8 matrix, except that there are eight rows, and each row contains eight elements.

Pixels 0 through n

Each of these pixel fields specifies a threshold level at which a particular pixel changes from black to white. Figure 7-41 shows the position of each pixel in a 4-by-4 matrix. The range of threshold values is from 0 to 255. In this range, 255 is the highest threshold setting, 0 is the lowest threshold setting, and 128 is the average threshold setting. A high threshold value results in most gray shades being changed to white. A low threshold value results in most gray shades being changed to black.

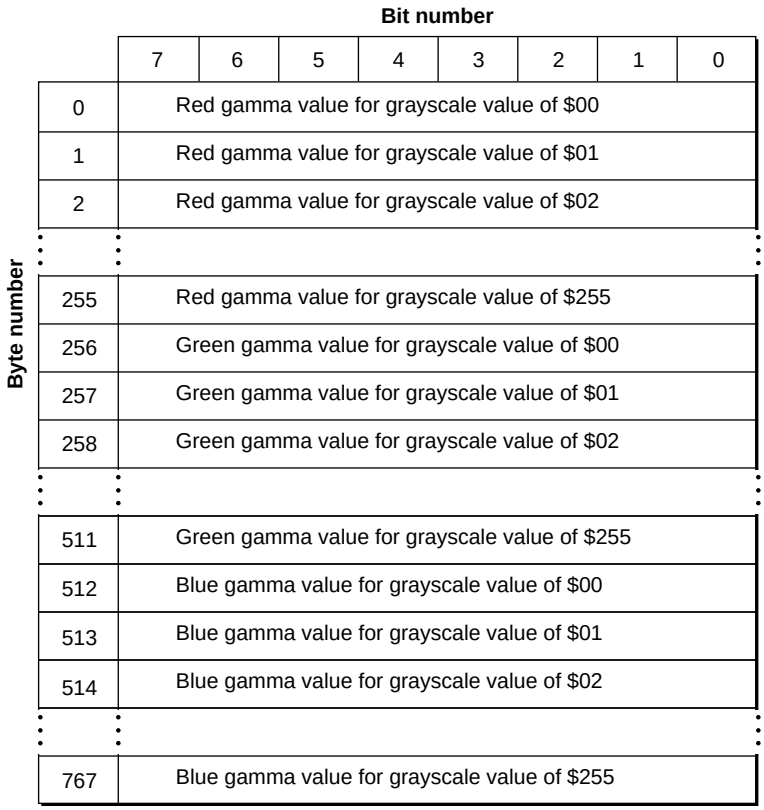
Figure 7-41 The halftone matrix pattern for a 4-by-4 matrix

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

Gamma Data Write Format for the Color OneScanner

Figure 7-42 shows the gamma data write format.

Figure 7-42 Gamma data write format



Color Correction Matrix for the Color OneScanner

Figure 7-43 shows the color correction matrix.

Figure 7-43 Color correction matrix

Byte number	Bit number 7	6	5	4	3	2	1	0
0	K0 byte 1 (Ka)							
1	K0 byte 2 (Kb)							
2	K1 byte 1 (Ka)							
3	K1 byte 2 (Kb)							
4	K2 byte 1 (Ka)							
5	K2 byte 2 (Kb)							
6	K3 byte 1 (Ka)							
7	K3 byte 2 (Kb)							
8	K4 byte 1 (Ka)							
9	K4 byte 2 (Kb)							
10	K5 byte 1 (Ka)							
11	K5 byte 2 (Kb)							
12	K6 byte 1 (Ka)							
13	K6 byte 2 (Kb)							
14	K7 byte 1 (Ka)							
15	K7 byte 2 (Kb)							
16	K8 byte 1 (Ka)							
17	K8 byte 2 (Kb)							

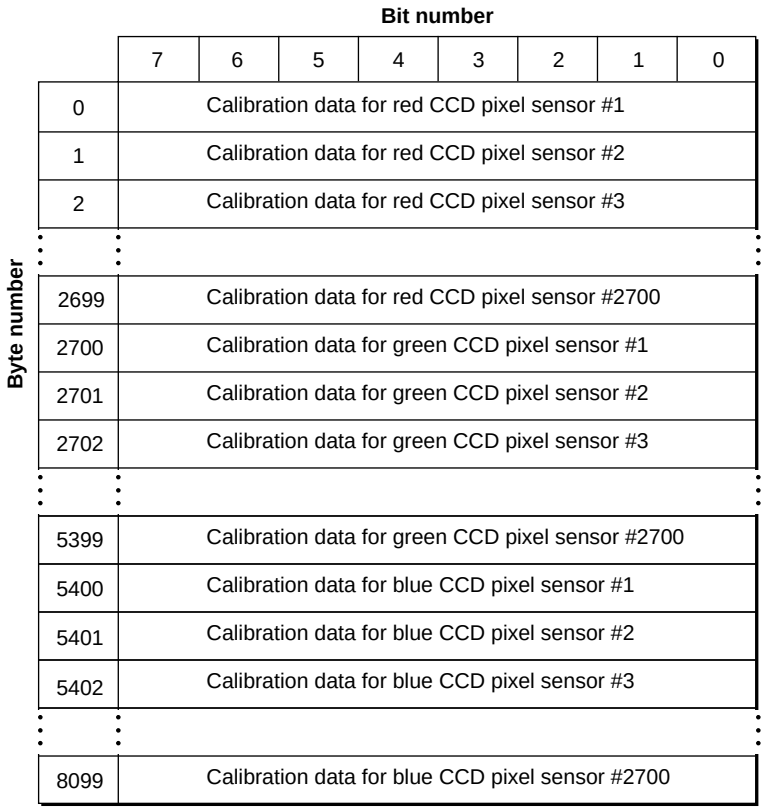
The format for each byte is shown below. Ka corresponds to byte 1 of K0-8, and Kb corresponds to byte 2 of K0-8. The sign bit indicates plus or minus, MSB is the most significant bit, LSB the least significant bit.

	Ka		Kb		
Byte	Bit 15	Bits 9-14	Bit 8	Bits 7-1	Bit 0
Ka and Kb	Sign	0	MSB	Data bits	LSB

PSRAM Data Write Format for the Color OneScanner

Figure 7-44 shows the PSRAM data write format.

Figure 7-44 PSRAM data write format



OBJECT POSITION ($31)

The OBJECT POSITION command allows the host computer to control the position of the scanner carriage assembly. The command provides options to park and unpark the carriage and to set the carriage to a specified position. Figure 7-45 shows the format of the OBJECT POSITION command structure.

Note
This command is valid only for the OneScanner and the Color OneScanner. The Apple Scanner does not support this command. ◆

Figure 7-45 The OBJECT POSITION command structure

Byte number	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	($31)			Operation code				
1	Reserved			Reserved		Position type		
2	($00)			Reserved				
3	Count (MSB)							
4	Count (LSB)							
5 – 9	($00)			Reserved				

FIELD DESCRIPTIONS

Here is a list of the fields used in this command structure:

Operation code
The operation code for the OBJECT POSITION command is $31.

Reserved These fields are reserved for future expansion. Set them to 0.

Position type This field specifies the action for the scanner to take. Valid values are as follows:

Unpark carriage $00
Instructs the scanner to move the carriage to the home position. If the carriage is already at the home position, the scanner returns a GOOD status. If the scanner cannot move the carriage to the home position, the scanner returns a CHECK CONDITION status and sets the sense data to VENDOR UNIQUE ($9).

Park carriage $01
Instructs the scanner to move the carriage to the shipment lock position. If the carriage is already at that position, the scanner returns a GOOD status.

Absolute position $02

Instructs the scanner to position the carriage at the scan line specified in the Count field. A 0 value causes the scanner to place the carriage at the beginning of the scan area. Other values must indicate a valid y-axis position in increments of 1/1200 of an inch. If the value of Count corresponds to an invalid scan line, the scanner returns a CHECK CONDITION status and sets the sense data to ILLEGAL REQUEST ($5).

Relative position $03

Instructs the scanner to move the carriage to a position relative to its current position. The value of the Count field indicates the direction and distance for the operation. Positive values move the carriage forward; negative values move the carriage backward. The distance is expressed in increments of 1/1200 of an inch. A 0 value does not move the carriage. If the value of Count yields an invalid scan line, the scanner returns a CHECK CONDITION status and sets the sense data to ILLEGAL REQUEST ($5).

Count — This field specifies the distance for absolute and relative position requests. The scanner ignores this field for park and unpark requests.

GET DATA STATUS ($34)

The GET DATA STATUS command allows the host computer to determine how much image data the scanner currently holds. The host computer can then decide whether to issue a READ command to retrieve the stored data. The scanner reports data availability for those windows that were defined with the DEFINE WINDOW PARAMETERS command and were specified in the current SCAN command. Figure 7-46 shows the format of the GET DATA STATUS command structure. The returned data blocks for each supported scanner are described later in this section.

Figure 7-46 The GET DATA STATUS command structure

Byte number	Bit number 7	6	5	4	3	2	1	0
0	($34)			Operation code				
1	($0)			Reserved				Wait
2-5	($00)			Reserved				
6	Allocation length (MSB)							
7	Allocation length							
8	Allocation length (LSB)							
9	Reserved							

FIELD DESCRIPTIONS

Here is a list of the fields and the bits used in this command structure:

Operation code
: The operation code for the GET DATA STATUS command is $34.

Reserved
: These fields are reserved for future expansion. Set them to 0.

Wait
: This bit indicates when the scanner returns data status to the host computer. If the value of the field is 1, the scanner waits for the quantity of data in the scanner's internal memory to reach the limit set by the MODE SELECT command before the scanner responds with data. (This limit is set by the Buffer full ratio field of the disconnect-reconnect parameter page.) If the value is 0, the scanner responds immediately.

Allocation length

This field specifies the number of bytes that the host computer has allocated for returned data status. If the value of the Allocation length is 0, the scanner will not return data status (this is not an error condition). With any other value, the maximum number of bytes is transferred. The command terminates when the bytes specified by the Allocation length field have been transferred, or when all available status data bytes have been transferred to the host, whichever comes first.

GET DATA STATUS Return Structure Description for the Apple Scanner

Figure 7-47 shows the format of the GET DATA STATUS return structure for the Apple Scanner.

Figure 7-47 The GET DATA STATUS return structure for the Apple Scanner

Bit number

Byte number	7	6	5	4	3	2	1	0
0	($00)				Data length (MSB)			
1	($00)				Data length			
2	($08)				Data length (LSB)			
3	($0)				Reserved			Block

Byte number	
0	Window identifier
1	($00) Reserved
2	($00) Reserved
3	($00) Reserved
4	($00) Reserved
5	Scan data available (MSB)
6	Scan data available
7	Scan data available (LSB)

FIELD DESCRIPTIONS

The parameter list header describes the length, in bytes, of the descriptor information that follows. Here are the parameter list fields used in this structure:

Data length — This field indicates the number of bytes of status data that follows, starting at byte 4. The value of this field does not include bytes 0 through 3. Each return structure is associated with a window descriptor. If the Apple bit is set in the DEFINE WINDOW PARAMETERS command, the scanner returns data status for only the first window descriptor. Data for secondary scan areas is returned together with the data for the primary scan area. The scanner indicates that a scan is complete and that all data has been transferred by setting the Data length field to 0 and sending no further status data.

Reserved — These fields are reserved for future expansion. Set them to 0.

Block — This field indicates the scanner buffer status. A value of 1 indicates that the scanner's internal buffers are full and that the scanner must transfer all available scan data to the host computer before it can generate more scan data. The bit is also set to 1 when the scanner has reached the end of the scan, even though the buffer may not be full. A value of 0 indicates that the scanner is not currently blocked because of insufficient buffer space.

The data status descriptors contain information about defined scan windows. Each descriptor applies to a single window. Here are the data descriptors used in this structure:

Window identifier — This field identifies the window associated with this return structure. This value matches the window identifier in the window descriptor parameter of the DEFINE WINDOW PARAMETERS command.

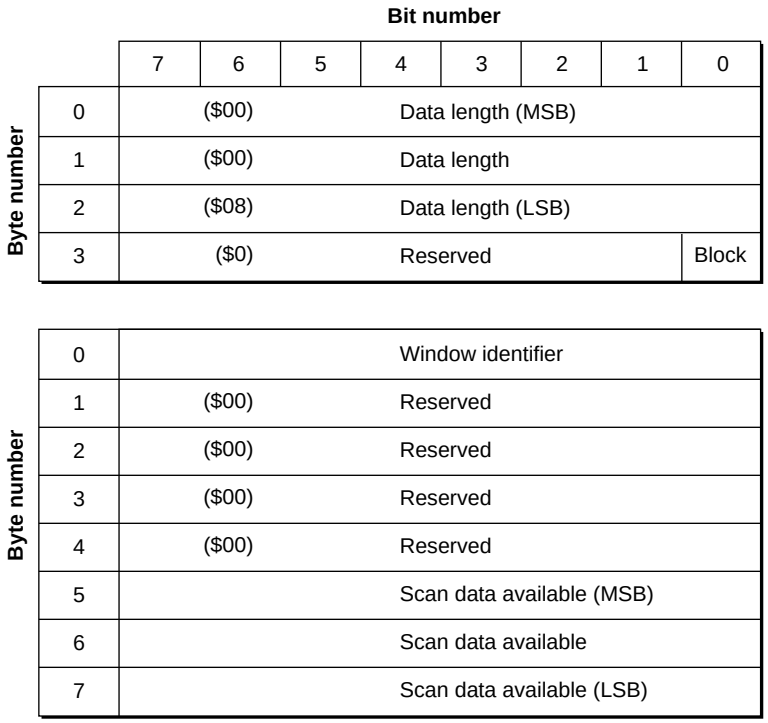
Reserved — These fields are reserved for future expansion. Set them to 0.

Scan data available — This field indicates the number of bytes of data that the scanner can send for the scan window identified by the Window identifier field. If the value is 0, it indicates that the scanner currently has no data available for the window (perhaps because it has not reached the window in its scanning process). This is not an error condition.

GET DATA STATUS Return Structure Description for the OneScanner and the Color OneScanner

Figure 7-48 shows the format of the GET DATA STATUS return structure.

Figure 7-48 The GET DATA STATUS return structure



FIELD DESCRIPTIONS

The parameter list header describes the length, in bytes, of the descriptor information that follows. Here are the parameter list fields and the bit used in this structure:

Data length — This field indicates the number of bytes of status data that follows starting at byte 4. The value of this field does not include bytes 0 through 3. The OneScanner supports a single scan window, so this command never returns more than one descriptor. The scanner indicates that a scan is complete and that all data has been transferred by setting the data length field to 0 and sending no further status data.

Reserved — These fields are reserved for future expansion. Set them to 0.

Block This bit indicates the scanner buffer status. A value of 1 indicates that the scanner's internal buffers are full and that the scanner must transfer all available scan data to the host computer before it can generate more scan data. The bit is also set to 1 when the scanner has reached the end of the scan, even though the buffer may not be full. A value of 0 indicates that the scanner is not currently blocked because of insufficient buffer space.

The data status descriptors contain information about defined scan windows. Each descriptor applies to a single window. Here are the data descriptors used in this structure:

Window identifier
This field identifies the window associated with this return structure. This value matches the window identifier in the window descriptor parameter of the DEFINE WINDOW PARAMETERS.

Reserved These fields are reserved for future expansion. Set them to 0.

Scan data available
This field indicates the number of bytes of data that the scanner can send for the scan window identified by the Window identifier field. A value of 0 indicates that the scanner currently has no data available for the window (perhaps because it has not reached the window in its scanning process). This is not an error condition.

Appendixes

Specifications

This appendix contains hardware specifications for the members of the Apple scanner product family.

Apple Scanner Specifications

Table A-1 contains the specifications of the Apple Scanner.

Table A-1 The Apple Scanner specifications

Specifications	
Physical proportions	
Depth	21.8 in. (545 mm)
Width	13.6 in. (340 mm)
Height	4.4 in. (110 mm)
Weight	20 lbs. (9.072 kg)
Microprocessor	
Type	8-bit NEC PD 7809
Timing	12 MHz
RAM size	256 bytes
ROM size	8 KB
DMAC	
Type	NEC PD8237
Timing	4 MHz
DIPP	
Type	NEC HD 63084
Timing	4 MHz
Dither pattern	4-by-4 matrix
Grayscale	16 gray shades
Memory	
RAM	32 KB
ROM	Two 2 KB by 8

continued

Table A-1 The Apple Scanner specifications (continued)

Specifications	
Noise (maximum)	
Standby	<30 dB
Scanning	<55 dB
Temperature	
Operating temperature	+10° C to +40° C
Storage (6 months)	–40° C to +47° C
Transit (72 hours)	–40° C to +65° C
Humidity	
Storage (6 months)	20% to 95% relative humidity
Power requirements	
AC input (U.S. & Canada model)	120 V AC ±10%, 58 to 62 Hz
AC input (Universal model)	100/120/200/220/240 V AC ±10%, 48 to 62 Hz
Power consumption	
Standby	35 watts
Scanning	65 watts

Apple OneScanner Specifications

Table A-2 contains the specifications of the Apple OneScanner.

Table A-2 The Apple OneScanner specifications

Specifications	
Physical proportions	
Depth	21.8 in. (545 mm)
Width	13.6 in. (340 mm)
Height	4.4 in. (110 mm)
Weight	23 lbs. (10.45 kg)
Microprocessor	
Type	8-bit 8031
Timing	12 MHz
RAM size	8 KB
ROM size	32 KB

continued

Table A-2 The Apple OneScanner specifications (continued)

Specifications	
Image processor	
Type	ASIC
Timing	12 MHz
Dither patterns	4-by-4 and 8-by-8 Spiral and Bayer patterns built in; downloaded patterns may range from 1-by-2 to 16-by-16
Grayscale	256 gray shades
Memory	
Calibration RAM	4 KB
RAM	32 KB
Noise (maximum)	
Standby	<30 dB
Scanning	<55 dB
Temperature	
Operating temperature	+10° C to +40° C
Storage (6 months)	–40° C to +47° C
Transit (72 hours)	–40° C to +65° C
Humidity (noncondensing)	
Storage (6 months)	20% to 95% relative humidity
Power requirements	
AC input (U.S. & Canada model)	120 V AC ±10%, 58 to 62 Hz
AC input (Universal model)	100/120/200/220/240 V AC ±10%, 48 to 62 Hz
Power consumption	
Standby	<22 watts
Scanning	<45 watts

Apple Color OneScanner Specifications

Table A-3 contains the specifications of the Apple Color OneScanner.

Table A-3 The Apple Color OneScanner specifications

Specifications	
Physical proportions	
Depth	21.8 in. (545 mm)
Width	13.6 in. (340 mm)
Height	4.4 in. (110 mm)
Weight	23 lbs. (10.45 kg)
Microprocessor	
Type	8-bit Z80
Timing	8MHz
RAM size	8KB
ROM size	64KB
Image processor	
Type	ASIC
Timing	16MHz
Dither patterns	Not supported
Grayscale	256 gray shades
Color	24-bit color
Memory	
Calibration RAM	4 KB
RAM	32 KB
Noise (maximum)	
Standby	<30 dB
Scanning	<55 dB
Temperature	
Operating temperature	+10° C to +40° C
Storage (6 months)	–40° C to +47° C
Transit (72 hours)	–40° C to +65° C
Humidity (noncondensing)	
Storage (6 months)	20% to 95% relative humidity

continued

Table A-3 The Apple Color OneScanner specifications (continued)

Specifications	
Power requirements	
AC input (U.S. & Canada model)	120 V AC ±10%, 58 to 62 Hz
AC input (Universal model)	100/120/200/220/240 V AC ±10%, 48 to 62 Hz
Power consumption	
Standby	<22 watts
Scanning	<45 watts

Optimizing the Color OneScanner

The Color OneScanner enables you to copy color images and transfer them into a Macintosh, or other computer. The goal is to have as close a match as possible between the color characteristics of the input image and the color characteristics of the output image, as displayed on the computer screen or as, perhaps, eventually printed out.

The quality of the output image is a function of the scanner's color matrix and the gamma correction matrix. This appendix describes additional programming extensions associated with these factors, which enable you to optimize the data contained in the scanned color image.

Optimizing the Scanner Matrix

Most color input scanners use three input channels to collect scanned data. These are the red (R), green (G), and blue (B) channels. The document being scanned (or copied) is typically illuminated with an approximately white light source. To produce the R, G, and B channels, filters are placed over the sensors that detect the reflected light through the scanner's optical system.

These separation filters are generally designed around a number of constraining parameters, such as signal-to-noise ratio, cost, and scanner speed. The spectral width of the filter is one of the factors that determines the quality of the color image.

If the RGB separation filters are too narrow, the document is sampled with too low a spectral width. This reduces the light level that reaches the detectors and may either impair the scanner's signal-to-noise ratio, or require that the scanning speed be slowed down, or both. If the filters are too broad, the R, G, and B signals overlap, seriously damaging the quality of the color image.

Typically, the spectral width of the filters is such that some portion of the spectral range covered overlaps the spectral range of the adjacent filter. For example, the blue-green side of the blue filter has some spectral overlap with the blue-green side of the green filter, and the green-red side of the red filter overlaps with the green-red side of the green filter. The red filter has no overlap with any spectral area of the blue filter, or vice versa.

This type of overlap causes certain spectral areas of a document, such as the blue-green and the green-red, to give rise to signals in two channels at the same time. If the spectral widths of the filters were sufficiently large, they would reach a point where the signals would completely overlap. The detrimental effect of overlapping, therefore, is limited by the narrowness of the spectral widths of the filters.

The overall effect of color overlapping represented by this matrix is shown in Table B-1.

Table B-1 Color overlap matrix

RED/red	RED/green	RED/blue
GREEN/red	GREEN/green	GREEN/blue
BLUE/red	BLUE/green	BLUE/blue

Names shown in upper-case letters (RED) indicate the filter color, and the names shown in lower case letters (green) indicate the other light color that comes through the filter. For example, RED/green indicates how much green light comes through the red filter, where RED is the filter and green is the spectral area considered. The matrix shown in Table B-1 indicates what happens to the signals when scanning occurs. This matrix can be inverted and scaled, resulting in negative off-diagonal terms that allow the filter effects to be corrected.

If you run the Color OneScanner with the matrix shown in Table B-2, you will obtain compensation for the filter effects and the scanner will produce much brighter colors. This is because compensation of the common filter spectral areas is properly utilized in the signals delivered into the red, green, and blue data.

Table B-2 Compensating for filter effects

1.1090	-0.1109	0.0019
-0.1428	1.1600	-0.0172
0.0296	-0.2075	1.1781

Gamma Correction

You must also consider the following factors when programming the Color OneScanner:

- If you scan a set of gray patches into the Color OneScanner, the channels do not respond ideally to the different shades of gray.
- The signals generated during the scan should track the reflectance characteristics of the samples scanned.
- The display screen on which the scanned image is viewed does not produce a luminance scale that tracks the reflectance scale of the scanned patches.

Scanner gamma correction values are intended to produce an optimized result when these factors are considered. Here is the gamma correction equation:

$Y = aX^g$ (1)

where
X is the input (representing the object scanned),
Y is the output (representing the image on the computer screen),
a is a constant, and
g is the gamma value.

The parameter *a* can be calculated from the following equation:

$a = 255 / (255^g)$ (2)

The input value from the scanner electronics is between 0 and 255. When this value is substituted in equation (1) it yields the output value. If g = 1, then a = 1, and Y tracks X, with a 1:1 correspondence. You are typically looking for a one-to-one match between the object scanned (input), and the image (output). If $g = 0.7$, then $a = 5.272$, and *Y* tracks *X* via equation (1), using the values derived from equation (2).

Another form of the equation for gamma can be shown by the following:

$Y = 255(x/255)^g$ (3)

You may set the gamma values, using the gamma matrix table. Table B-3 shows the gamma values used for a Rasterops 19-inch monitor, and for an Apple 13-inch RGB monitor.

Table B-3 Gamma values for Rasterops and Apple monitors

Monitor	Card	Red gamma	Green gamma	Blue gamma
Apple 13" RGB	8-24GC	0.609	0.578	0.641
Rasterops 19"	24XLTV	0.732	0.700	0.763

The Apple monitor was set to a gamma value of 2.2, and the Rasterops monitor to a gamma value of 1.8. You may prefer different values, but, in general, the ones indicated should work as described.

Different values for the different colors are appropriate, as shown in Table B-3. However, you should notice that the ratios of red to green, and green to blue are approximately the same for each monitor. If you change the gamma value, you should keep the ratios between the colors the same to control the proper balance of color from the scanner.

Glossary

American National Standards Institute (ANSI) A committee that sets protocol and measurement standards.

application program The program executing in the host computer. This program takes input from the user in the form of mouse movements, button clicks, and keyboard text and performs tasks in response to this input.

automatic background adjustment A feature of the Apple Scanner that allows more detail to be brought out from the dark areas of the original document by automatically adjusting the brightness level as the scanner scans dark areas of the document.

Bi-level Color mode A composition mode that produces 8-color bitmaps. Although the scanner scans in color, each color component (red, green, and blue) has only two possible values: on and off.

bitmap A set of bits that represents the graphic image of an original document in memory.

brightness parameter The parameter that determines the overall whiteness of the image during a halftone or grayscale scan. Increasing brightness results in a lighter overall image. Decreasing brightness results in a darker overall image.

buffer A "holding area" of the computer's memory where information can be stored by one program or device and then read at a different rate by another—for example, a print buffer.

carriage The light-sensitive scanning mechanism, including the lamp, that moves along the scanner bed during a scan.

CCD Charge-coupled device made up of an array of coupled capacitors. It is commonly used in imaging applications.

color correction Changes the color data obtained from one source for output to a destination device. One objective of color correction might be to correct the data so that, when it is displayed, it matches an original photograph. A hardware mechanism called the 3-by-3 matrix multiplier implements the color correction process for the Color OneScanner.

composition parameter The parameter that instructs the scanner whether to process a scan area as line art, halftone, or grayscale. See also ***Line Art mode, Halftone mode, Grayscale mode, Bi-Level Color mode,*** and ***Full Color mode.***

contrast parameter The parameter that condenses or expands the range between black and white for Halftone mode or Grayscale mode. A high contrast records different shades of gray, emphasizing blacks and whites. A low contrast records different shades of gray, emphasizing middle gray shades at the expense of blacks and whites.

data Any representation, such as characters or analog quantities, to which meaning is assigned. In the Apple scanners, image data consists of black and white (or gray) dots, representing dots on the original document.

dot See ***pixel.***

driver See ***scanner driver.***

fluorescent lamp See ***lamp.***

Full Color mode A composition mode that generates the largest range of colors. The scanner outputs color pixel data for each line, in three planes: red, green, and blue. Each of the three color components has 256 possible values (8 bits of each color), for a total of 16.8 million colors. Three colors with 8 bits each produces 24-bit color depth.

gamma correction A correction technique that compensates for loss of detail in dark objects.

graymap parameter The parameter that determines the relationship between the 16 or 256 levels of gray that the scanner detects and the actual range of gray shades between black and white in the original. You can set the graymap parameter to bring out more light detail, more dark detail, or to show only normal detail.

Grayscale mode A composition mode that instructs an application program to record all shades of gray detected by the scanner. The resulting grayscale image can then be displayed on a grayscale monitor or converted to halftone or line art by the program.

Halftone mode A composition mode that uses combinations of black and white dots to represent shades of gray. In the halftone process used in printing, the number, pattern, size, and shape of the dots can be varied. In electronic processing, only the number and pattern of the dots can be varied. The human eye perceives a close approximation between a good halftone representation and the original shades of gray. Use Halftone mode to scan photographs, drawings, and other originals so that the image shows shades of gray. See also ***halftone pattern.***

halftone pattern The matrix of threshold values that determine the patterns of black and white dots that represent different shades of gray in a Halftone mode scan area. You can change this pattern using any combination of 16 threshold values.

host computer Any microcomputer equipped with a SCSI port and capable of sending commands and parameters to and retrieving image data from the scanner.

image The result, residing in memory, of scanning an original. Also known as the *scan image*.

information transfer phases See ***transfer phases.***

lamp The special fluorescent light used by the scanner to illuminate the original during a scan. The light reflected from the original is detected by the scanner and recorded as electronic information.

Line Art mode A composition mode that instructs the scanner to record each dot on the original as either a black or white dot. Use Line Art mode to scan text and line drawings that contain no gray shades. If you scan an original that contains gray shades, the brightness parameter determines the level at which a gray shade is recorded as black or white.

optical character recognition A process by which text on paper can be scanned and converted into text files on a computer.

original What you place on the scanner glass. The original can be a document, a photograph, or an object, such as a coin, that you want to scan.

parameter A variable that is assigned a value and then provided in the parameter list of a function or command when calling or invoking the function or command.

pixel Short for *picture element*. The size of a pixel is determined by the scan resolution. At a resolution of 300 dots per inch, 300 pixels are detected in each vertical inch and each horizontal inch. The term is used interchangeably with *dot*.

representative area A small portion of the image that you can use to test changes to any settings. You can display the new results on the screen or print them.

resolution A measure of the ability to delineate visual detail, which is usually specified in dots per inch (dpi). The higher the value, the finer the detail of the image.

resolution parameter A parameter that determines how finely the scanner scans the original, expressed in dots per inch. The resolution parameter determines the number of dots detected horizontally and vertically.

scan area parameter The parameter that specifies an area within the 8.5-by-14-inch scanner glass indicating the size and location of the area to scan. With the Apple Scanner, you can define one or more scan areas for each scan. The scan area is also known as the *window.*

scanner Any graphic input device that converts printed matter into bit (binary digital) data.

scanner driver Software that recognizes a predefined set of functions and, in turn, instructs the Apple scanners to perform the individual tasks that result in a scanned image.

scanner glass The glass surface that makes up the top surface of the scanner. This glass surface is where the user places the original document to be scanned.

scanning The process of digitizing an image for use with a computer.

sense data Sense data is status information transferred from the scanner to the host. It consists of a number of ***Sense keys*** .

Sense key A Sense key is a subset of ***sense data***. Each key indicates a different status or error condition. For example, Sense key $4 indicates a hardware error, Sense key $2 indicates that the scanner is not ready for operation.

Small Computer System Interface (SCSI) A standard interface developed to permit extremely high-speed data transfer among peripheral devices and a personal computer.

threshold parameter The parameter that affects the operation of Automatic Background Adjustment in the Apple Scanner.

transactions The host and scanner communicate with defined commands, transmitted in packets. These message exchanges are called transactions.

transfer phases Refers to phases (command, data, status, and message) during which data or control information is transferred between devices on the SCSI bus.

window See ***scan area parameter.***

Index

Numerals

3-by-3 matrix multiplier 9

A

ABA. *See* automatic background adjustment
aborting scan 33
ABORT message 109
absolute position
  moving carriage to 64, 185
  setting 59, 150
accessing scanner 16
Acknowledge signal (SCSI) 105
ACK signal 105
advanced capabilities, determining 27–29, 46
advanced constants 92–93
advanced data structures 83–90
advanced data types 95
advanced functions 14, 45–65
  determining those supported 86
  listed 96
  `ScGetAdvFeatures` 46
  `ScGetButton` 47
  `ScInvertPixels` 48
  `ScLoadGamma` 49
  `ScLoadMatrix` 50
  `ScResetButton` 51
  `ScSensorSelect` 52
  `scSensorSelect` 52
  `ScSetGraymap` 53
  `ScSetGroup3` 54
  `ScSetHTPattern` 55
  `ScSetLamp` 56
  `ScSetLed` 57
  `ScSetNoCal` 58
  `ScSetNoHome` 59
  `ScSetScannerAtoD` 60
  `ScSetSpeed` 61
  `ScSetThreshold` 62
  `ScSetWaitButton` 63
  `ScVendorUnique` 64
advanced scanner features, using 27
Apple Programmers and Developers Association (APDA) xvi
Apple Scanner 12, 14
  automatic background adjustment 10
  bits per pixel values 159
  brightness values 158
  contrast values 158
  data compression 159, 163, 166
  features 12
  grayscale support 5
  halftone patterns supported 159
  multiple scan area support 8
  resolutions supported 156
  SCSI hardware interface 104–107
  SCSI messages 108
  SCSI sense data 115
  specifications 193
Apple-specific MODE SELECT parameter page
  Apple Scanner 134
  Color OneScanner 137
  OneScanner 135
Apple-specific MODE SENSE data page
  Apple Scanner 144
  Color OneScanner 147
  OneScanner 145
applications, scanning 10–12
Arbitration, SCSI bus phase 106
assembly language, using scanner driver from
  scanner driver, using from assembly language 29
ATN signal 105
Attention signal (SCSI) 105
automatic background adjustment
  defined 10
  determining if available 86
  reading threshold 145
  setting threshold 62, 134, 158

B

backward compatibility 14
`badUnitErr` result code 98
basic scanning parameters 4
beginning a scan 11
Bi-level Color mode 4
bitmaps 3
bits per pixel 159, 162, 166
  specifying 73
  valid values 77
blue gamma field 89
brightness 4, 158, 162, 165
  getting valid values (example) 19–22
  specifying 71
  valid values 75, 76
`brightnessMax` field of `ScCompRec` record 62
brightness parameter 7
BSY signal 105
buffer full ratio 139
bus contention, resolving 140
BUS DEVICE REJECT message 109
Bus Free, SCSI bus phase 106
bus phases, SCSI 106–107
Busy signal (SCSI) 105
button, scanner
  reading 47, 136, 138, 146, 148
  resetting 51, 136, 137, 146, 148
  waiting for 63, 150
`byteWidth` parameter to `ScDoScan` function 36, 54

C

calibration, defeating 58, 151
calibration RAM
  reading 65, 170
  writing 64, 176
canceling scan 33
capabilities, determining scanner 42
carriage assembly 2, 59, 150
  moving to absolute position 64, 185
  moving to home position 64, 184
  moving to relative position 64, 185
  parking 64, 184
  positioning 59, 150

carriage assembly *(continued)*
returning to home position after scan 59, 150
CCD (charge-coupled device) array 2
power to 136, 146
CCITT Group III 54
C/D signal 106
CHECK CONDITION status code 108
close, Device Manager routine 34
closing the scanner 16–19, 34
color correction 8
color correction matrix 182
Color mode and brightness 7
Color OneScanner 12, 14
features 12
optimization 199
color overlap matrix 200
color overlapping 199
Command, SCSI bus phase 106
COMMAND COMPLETE message 108, 109
command structures
GET DATA STATUS ($34) 186–190
INQUIRY ($12) 122–132
MODE SELECT ($15) 133–139
MODE SENSE ($1A) 143–148
OBJECT POSITION ($31) 184–185
READ ($28) 170–175
RELEASE ($17) 142
RESERVE ($16) 140–141
SCAN ($1B) 149–152
SEND ($2A) 176–183
SEND DIAGNOSTIC ($1D) 153
TEST UNIT READY ($00) 113
compensating for human vision 8
composition
defined 4
determining modes supported by scanner 19–23, 82
mode, attributes of a 75
setting 154
specifying 73
compression, image data 54, 159, 163, 166
configuration, reading scanner 143
configuring scanner 133
contiguous memory, reading image data into 36
contrast 4, 5, 158, 162, 165
getting valid values (example) 19–23
specifying 72
valid values 75, 77
contrast parameter 5
control, Device Manager routine
`ScGetButton` function 47
`ScInvertPixels` function 48
`ScLoadGamma` function 49
`ScLoadMatrix` function 50
`ScResetButton` function 51
`ScSensorSelect` function 52
`ScSetGraymap` function 53
`ScSetGroup3` function 54
`ScSetHTPattern` function 55
`ScSetLamp` function 56
`ScSetLed` function 57
`ScSetNoCal` function 58
`ScSetNoHome` function 59
`ScSetScanArea` function 44
`ScSetScannerAtoD` function 60
`ScSetSpeed` function 61
`ScSetThreshold` function 62
`ScSetWaitButton` function 63
`ScVendorUnique` function 64
Control/Data signal (SCSI) 106

D

dark detail, enhancing 53, 134, 144
Data, SCSI bus phase 106
data bits, packing of returned 73
data buffer 11
data compression 159, 163, 166
Data In, SCSI bus phase 106
data integrity 61, 136, 138, 146
Data Out, SCSI bus phase 106
data structures 68–90
advanced 83–90
`ScMatrix` 90
standard 68–83
data transfer speed, controlling 61, 136, 138, 146
DEFINE WINDOW PARAMETERS command 134, 145, 152, 154
DEFINE WINDOW PARAMETERS command structure 154, 155
DEFINE WINDOW PARAMETERS parameter list 156, 159, 163
defining image buffer 11
determining scanner capabilities 14, 19–24
device driver. *See* scanner driver
Device Manager 29
Device Manager function equivalents 100
diagnostics, running scanner 153
dimensions, supported halftone pattern 85
direction, scan 162
DISCONNECT message 109
disconnect-reconnect parameter page 133, 138
downloading halftone patterns 176
driver name 16, 29
driver. *See* scanner driver

E

enhancing dark detail 8, 53, 134, 144
enhancing light detail 8, 53, 134, 144
error codes
hardware 114
scanner driver 97
error reporting, SCSI behavior 107
examples 16–29
excess pixels, masking off 70
existing bitmaps, reading into 70

F

FAX encoding of image data 54
features
determining scanner 42
features, querying scanner 122
file formats 11
filter effects, compensating for 200
fluorescent lamp 2, 58, 151
controlling 56, 135, 136, 138, 145, 146
Full Color mode 4, 5
function result codes, scanner driver 97

G

gamma correction 8, 53, 134, 144, 200
gamma data write format 181
gamma values
setting for different monitors 201
GET DATA STATUS command 170, 186

GET DATA STATUS return structure
Apple Scanner 187
Color OneScanner 189
OneScanner 189
GOOD status code 108
graphics file formats 11
graymap 8
setting 53, 134, 144
Grayscale mode 4, 5
and brightness 5, 7, 72
and contrast 5, 72
green gamma field 89

H

`halfToneElements` field of `ScCompRec` record 40
`halftone` field of `ScAreaRec` record 40, 55, 78
halftone matrix pattern 180
Halftone mode 4, 40, 55
and brightness 4, 7, 72
and contrast 4, 72
downloaded patterns 87
halftone parameter page description 179
halftone patterns 4, 159, 162
defining 87
determining those supported 40
determining valid patterns (example) 19–23
dimensions of downloaded patterns 86
dimensions supported 85
downloading 55, 85, 176
names of those supported 78
reading downloaded patterns 170
restrictions on downloaded patterns 85
specifying 73
hardware error codes 114
hardware reset, SCSI behavior 107
high-speed scanning 61, 136, 138, 146
home position, moving carriage to 64, 184
human vision and scanned images 8

I, J

IDENTIFY message 109
image buffer 11
image data, reading (example) 24–27
information transfer phases, SCSI bus 107
initializing scanners 11
initializing scanning parameters 11
Input/output signal (SCSI) 106
INQUIRY command 122, 170
INQUIRY command return structure
Apple Scanner 123
Color OneScanner 129
OneScanner 126
intensity curve 49
interface compatibility 14
I/O signal 106

K

`killIO`, Device Manager routine 33

L

lamp, controlling 56, 135, 136, 138, 145, 146
LED, controlling 57, 136, 137, 146, 148
light detail, enhancing 53, 134, 144
Line Art mode 4, 62
and brightness 4, 7, 71

M

matrix multiplier 9
Message, SCSI bus phase 107
Message In, SCSI bus phase 107
Message Out, SCSI bus phase 107
MESSAGE REJECT message 109
messages, SCSI 108–109
Message signal (SCSI) 106
minimum read size 35, 77
`minReadSize` field of `ScStdFeaturesRec` record 35
modes
Bi-level Color 4, 5
Full Color 5
Grayscale 5
Halftone 5
Line Art 4
MODE SELECT command 133
MODE SENSE command 143
MSG signal 106

N

`noErr` result code 98
noncontiguous memory, reading image data into 36

O

OBJECT POSITION command 184
OneScanner 12, 14
bits per pixel values 162, 166
brightness values 162, 165
contrast values 162, 165
features 12
grayscale support 5
halftone patterns supported 162
resolutions supported 161, 163
scan direction 162
SCSI hardware interface 104–107
SCSI messages 109
SCSI sense data 117, 119
specifications 194
open Device Manager routine 43
`openErr` result code 98
opening the scanner 16–19, 43
optical character recognition (OCR) 2
optimizing the Color OneScanner 199
orchestrating a scan 10–12

P

packing of returned data bits 73
parameters, setting scanner 44
parking carriage assembly 64, 184
PICT file format 11
pin assignments, SCSI connector 104

pixel, bits to represent one 73
pixels and resolution 3
pixels per inch 7
PNTG file format 11
position, moving carriage to 64, 185
positioning carriage assembly 59, 150
power signal (SCSI) 105
power-up, SCSI behavior 107

Q

querying scanner capabilities (example) 19–23
quitting scan 33

R

read, Device Manager routine 35
READ command 170
READ command structure
  Apple Scanner 170
  Color OneScanner 173
  OneScanner 171
reading calibration RAM 170
reading downloaded halftone patterns 170
reading into existing bitmaps 70
reading scanner data 11, 35, 77, 170, 186
  example 24–27
reading static RAM 170
read size, minimum 77
ready status, checking 113
rectangle, reading into 36
red gamma field 89
relative position, moving carriage to 64, 185
RELEASE command 142
releasing the scanner 16–19, 34, 142
REQ signal 106
REQUEST SENSE command 113, 114
REQUEST SENSE return structure
  Apple Scanner 115
  Color OneScanner 119
  OneScanner 117
Request signal (SCSI) 106
Reselection, SCSI bus phase 106
resElements field of ScCompRec record 41
RESERVATION CONFLICT status code 108
RESERVE command 140
reserving the scanner 16–19, 43, 140
Reset signal (SCSI) 105
resolution 3
  1-dpi increment support 76
  Apple Scanner values 156
  determining supported 41
  getting valid values (example) 19–23
  OneScanner values 161, 163
  setting 154
  setting horizontal and vertical differently 86
  specifying 69
  valid values 79
resolution parameter 7
resolving bus contention 140
result codes
  badUnitErr 98
  noErr 98
  openErr 98
  scBusy 99
  scComErr 98
  scDimLampErr 99
  scEOS 99
  scLampErr 99
  scNotFoundErr 98
  scParamErr 98
  scResetErr 98
  scScannerErr 98
  statusErr 98, 99
result codes, scanner driver 97
retrieving scanned data 11
rowBytes parameter to ScDoScan function 37, 54
RST signal 105

S

sample code 16–29
  opening the scanner driver 17
  retrieving information 20
  setting up scan parameters 23
  starting the scan 25
  using advanced features 28
  valid resolution 22
ScAbortScan function 33, 35
ScAdvFeaturesRec record 46
  defined 84
scan, starting (example) 24–27
scan area 8, 24
  defining 80
  dimensions 81
  setting (example) 24
  setting dimensions 154
  specifying 69
scan areas, secondary 80, 84
SCAN command 149
SCAN command structure
  Apple Scanner 149
  Color OneScanner 151
  OneScanner 150
scan direction 162
scanned image, defining 8
scanner capabilities 19
scanner components 2
scanner data, reading 24–27
scanner driver
  advanced constants 92–93
  advanced data types 95
  advanced functions 14, 15, 45–65
    listed 96
  advanced functions supported 86
  applications 14
  assembly-language programs 29, 100
  closing 34
  data structures 68–??
  determining advanced capabilities 46
  error codes 97
  introduction to 14
  name 16, 29
  opening 43
  sample code 16–29
  standard constants 92
  standard data types 93–94
  standard functions 14, 15, 32–??, 96
  using 16–29
  version 81, 84
scanner drivers 14
scanner features, querying 122
scanner intensity curve 89
scanner matrix, optimizing 199
scanners and human vision 8
scanner-specific features, using 64
scanner-specific functions 14
scanner type identifiers 81
scanning application functions 10
scanning applications 10–12
scanning area 3, 8
scanning parameters 4, 11
  brightness 7

color correction 8
scanning parameters *(continued)*
composition 4
contrast 5
gamma correction 8
resolution 7
scan area 8
threshold 10
scanning parameters, setting 44, 154
scanning process 2, 10–12
scan parameters, setting 23, 44
scan speed, controlling 61, 136, 138, 146
scan time, reducing
by controlling lamp 56, 135, 136, 138, 145, 146
by defeating calibration 58, 151
by positioning carriage 150
with ScSetNoHome function 59
ScAreaRec record
and ScScanAreaRec record 80
defined 69
scBusy result code 99
ScClose function 16–19, 34, 56
scComErr result code 98
ScCompRec record
and ScStdFeaturesRec record 81
defined 75
scDimLampErr result code 99
ScDoScan function 24–27, 33, 35
scEOS result code 99
ScGetAdvFeatures function 27–29, 46, 84
ScGetButton function 47, 51, 86
ScGetHalfTones function 19, 23, 40, 78
ScGetRes function 19–23, 41, 79
ScGetStdFeatures function 42, 44, 75, 81
example 19–23
ScHalfToneArray record 40
ScHalfToneArray structure
defined 78
size of 76
ScInvertPixels function 48, 86
scLampErr result code 99
ScLoadGamma function 49, 86
ScLoadGamma record function 89
ScLoadMatrix function 50
ScMatrix 90
ScMatrixMul function 86
scNotFoundErr result code 98
ScOpen function 16–19, 43
scParamErr result code 98
ScPatRec record 55, 87
ScResArray record 41, 79
ScResArray structure, size of 76
ScResetButton function 47, 51
scResetErr result code 98
ScScanAreaRec record 44, 80
scScannerErr result code 98
ScSensorSelect function 52, 86
ScSetAtoD function 86
ScSetAto function 86
ScSetGraymap function 53, 86
ScSetGroup3 function 54, 86
ScSetHTPattern function 55, 85, 87
ScSetLamp function 56, 86
ScSetLed function 57, 86
ScSetNoCal function 58, 86
ScSetNoHome function 59, 86
ScSetScanArea function 24, 35, 44, 69, 80
restrictions on parameters 86
ScSetScannerAtoD function 60
ScSetSpeed function 61, 86
ScSetThreshold function 62, 86
ScSetWaitButton function 63, 86
SCSI bus phases 106–107
SCSI commands 112–190
DEFINE WINDOW PARAMETERS ($24) 154
GET DATA STATUS ($34) 186
INQUIRY ($12) 122
listed 112
MODE SELECT ($15) 133
MODE SENSE ($1A) 143
OBJECT POSITION ($31) 184
READ ($28) 170
RELEASE ($17) 142
REQUEST SENSE ($03) 114
RESERVE ($16) 140
SCAN ($1B) 149
SEND ($2A) 176
SEND DIAGNOSTIC ($1D) 153
status codes 108
TEST UNIT READY ($00) 113
SCSI commands and operation codes 112
SCSI hardware interface 104–107
SCSI messages 108–109
SCSI signals 105–106
SCSI status codes 108
ScStdFeaturesRec record 42
defined 81
ScVendorUnique function 64
secondaryMax field of ScAdvFeaturesRec record 80
secondary scan areas 80, 84, 156, 166
and composition mode 86
Selection, SCSI bus phase 106
Select signal (SCSI) 106
self-test, running scanner 153
SEL signal 106
SEND command 176
SEND command structure
Apple Scanner 176
Color OneScanner 178
OneScanner 177
SEND DIAGNOSTIC command 153
sense data, retrieving SCSI 114
Setting 23
setting calibration RAM 176
setting scan parameters 23
setting static RAM 176
signals, SCSI 105–106
specifications, of Apple scanners 193–195, ??–197
speed, controlling scan 61, 136, 138, 146
standard capabilities, determining 19–23
standard constants 92
standard data structures 68–83
standard data types 93
standard features of scanner, determining 42
standard functions 14, 32–44
listed 96
ScAbortScan 33
ScClose 34
ScDoScan 35
ScGetHalfTones 40
ScGetRes 41
ScGetStdFeatures 42
ScOpen 43
ScSetScanArea 44
starting a scan 11, 24–27, 35, 149
startup, SCSI behavior 107
static RAM
reading 170
writing 176
status, Device Manager routine 40, 41, 42, 46
status, retrieving scanner 114
Status, SCSI bus phase 106
status codes, SCSI 108
statusErr result code 98, 99
storing scanned data 11

summary of scanner driver 92–101

T

Terminator power signal (SCSI) 105
TERMPWR signal 105
TEST UNIT READY command 113
threshold 62, 158, 162, 165
  automatic adjustment 10
threshold parameter 10
transfer data types 174
transfer identification 174
type identifier, scanner 81

U

using the scanner driver 16–29

V

vendor-unique features, using 64
version, driver 81, 84
`version` field of
    `ScAdvFeaturesRec` record 46
`version` field of
    `ScStdFeaturesRec` record 42

W, X, Y, Z

window identifier byte 152
writing calibration RAM 176
writing scanner data 176
writing static RAM 176

This Apple manual was written, edited, and composed on a desktop publishing system using Apple Macintosh computers and FrameMaker software. Proof pages were created on an Apple LaserWriter IINTX printer. Final pages were created on the Varityper VT600 imagesetter. Line art was created using Adobe™ Illustrator. PostScript™, the page-description language for the LaserWriter, was developed by Adobe Systems Incorporated.

Text type is Palatino® and display type is Helvetica®. Bullets are ITC Zapf Dingbats®. Some elements, such as program listings, are set in Apple Courier.

WRITER
Joyce D. Mann

DEVELOPMENTAL EDITORS
Antonio Padial, Beverly Zegarski

ILLUSTRATOR
Deborah Dennis

PRODUCTION EDITOR
Rex Wolf

Special thanks to Shirley Foreman, Steve Bischoff, Lin Chan, Forrest Tanaka, Alex Pollayil, and Rolly Reed

Acknowledgments to Doug Engfer, who wrote the first edition of this book.