

Cortland Toolbox Reference: Volume I

Alpha Draft
Part No. 030-3123-A
June 10, 1986

Writer: William H. Harris
Apple Technical Publications

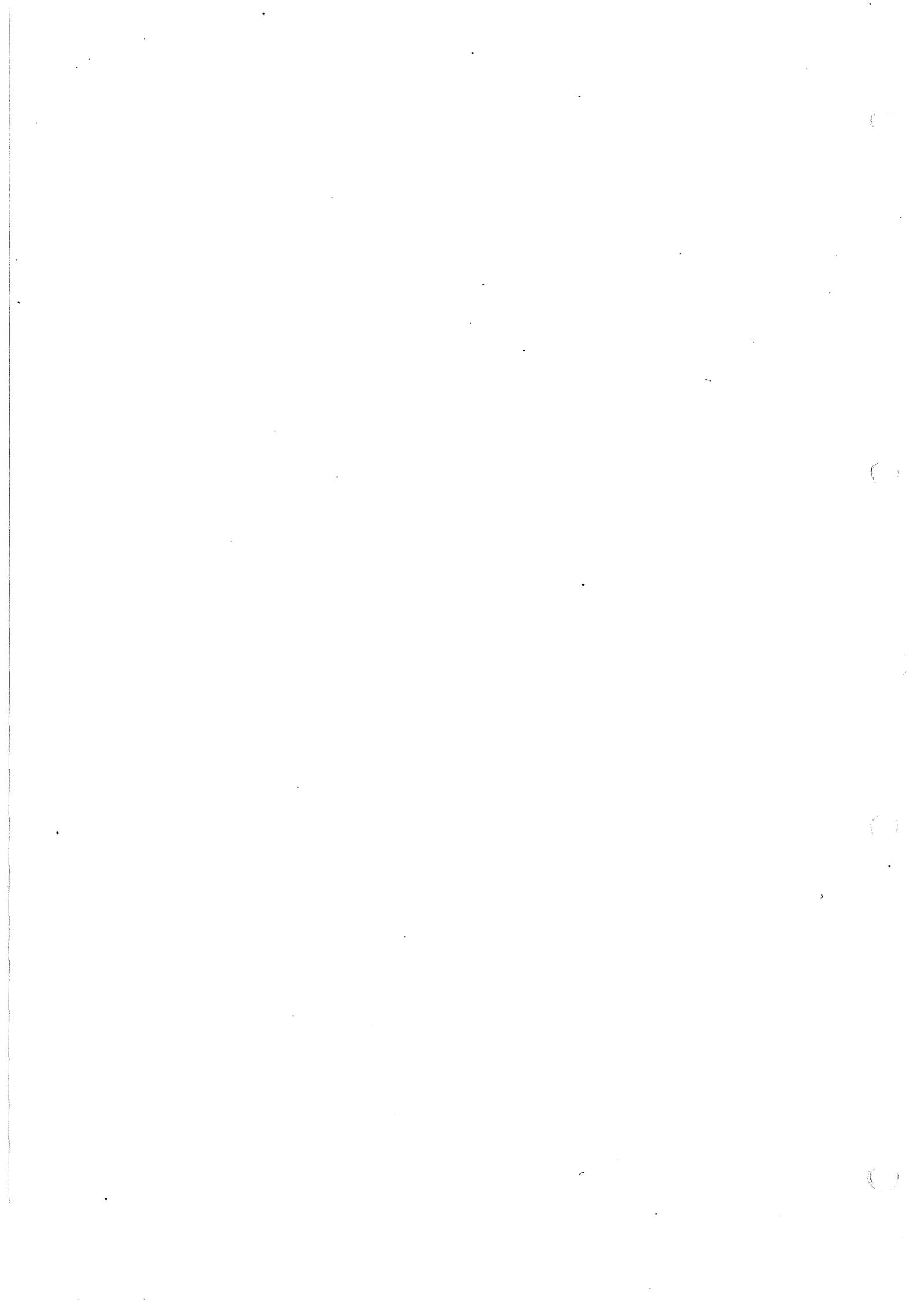
Changes Since Last Draft

This is the first draft of this document.

Sources

This document is based on the following ERS's:

Integer Math Tools	V00:20	May 9, 1986
Event Manager	00:40	March 4, 1986
Tool Locator		February 19, 1986
Memory Manager	Rev 4	March 10, 1986
Menu Manager		May 8, 1986
Miscellaneous Tools	Ver 0.86	May 9, 1986
QuickDraw II		June 3, 1986
Cortland SANE Tool Set	0.02	12 December 1985
Sound Tools	Rev. 1.4	June 6, 1986
Text Tools	Ver 0.16	May 27, 1986





Preface

About This Manual

This manual provides the specifications of to the Cortland Tools for the application programmer. It defines the terms used in describing the tools and provides background information. .

Roadmap to the Cortland Technical Manuals

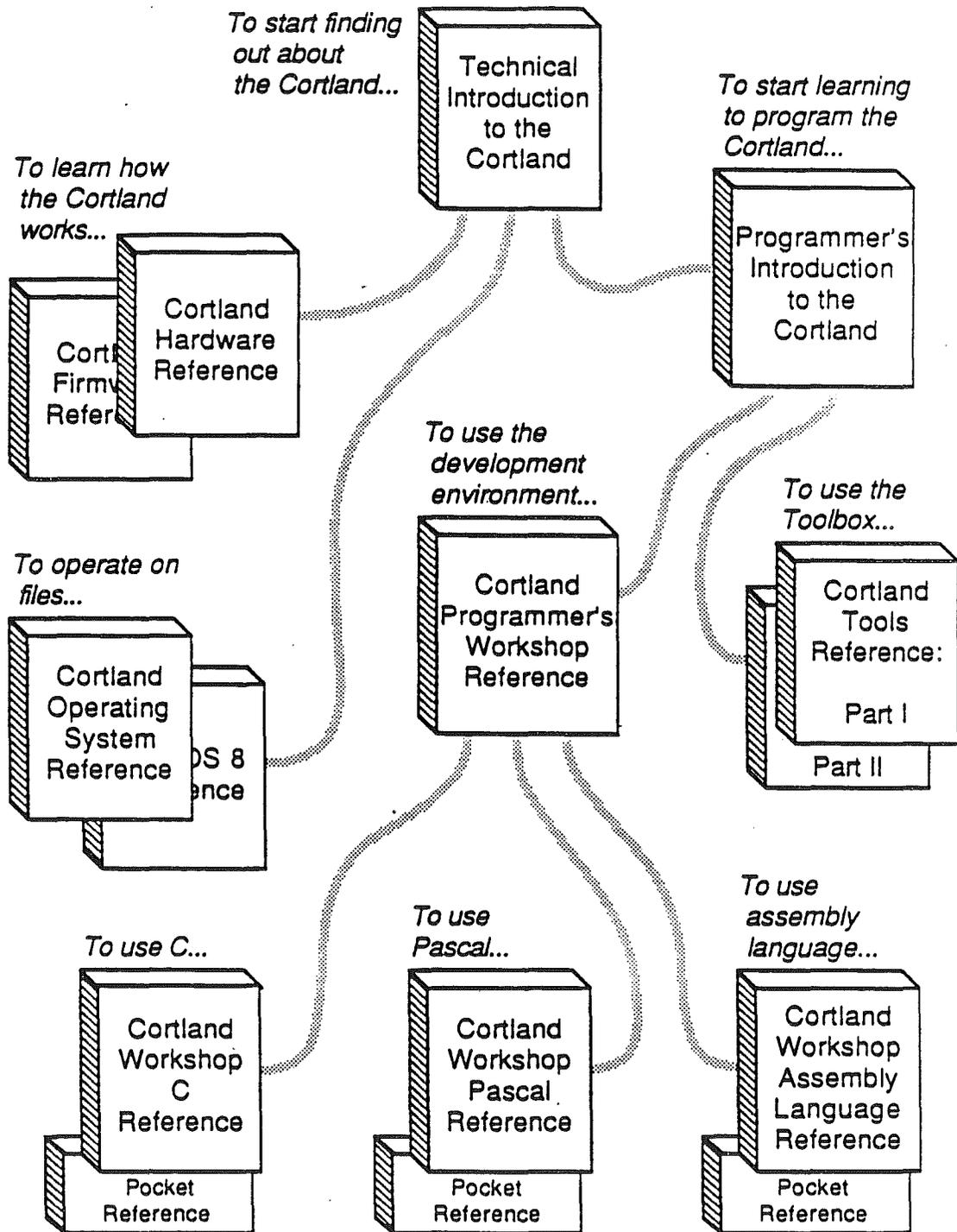
The Cortland has many advanced features, making it more complex than earlier models of the Apple II. To describe it fully, Apple has produced a whole suite of technical manuals. The manuals are listed in Table A-1. Figure A-1 is a diagram showing the relationships among the different manuals. Depending on the way you intend to use the Cortland, you may need to refer to a select few of the manuals, or you may need to refer to most of them.

Table A-1. The Cortland Technical Manuals

Title	Subject
Technical Introduction to the Cortland	what the Cortland is
Cortland Hardware Reference	machine internals—hardware
Cortland Firmware Reference	machine internals—firmware
Writing Cortland Programs	sample program using the toolbox
Cortland Toolbox Reference: Volume I	toolbox specifications
Cortland Toolbox Reference: Volume II	more toolbox specifications
Cortland Programmer's Workshop	the development environment
Cortland Workshop Assembly Language Reference*	using assembly language
Cortland Workshop C Reference*	using C on the Cortland
Cortland Workshop Pascal Reference*	using Pascal on the Cortland
ProDOS/8 Technical Reference	ProDOS for Apple II programs
Cortland Operating System Reference	ProDOS and loader for Cortland
Human Interface Guidelines	for all Apple computers
Apple Numerics Manual	numerics for all Apple computers

*There is a Pocket Reference for each of these.

Figure A-1. Roadmap to the technical manuals



The Technical Introduction

text

The *Technical Introduction to the Cortland* tells a little about a lot of things, but it doesn't tell everything about anything. To find out all about any one aspect of the Cortland, you should read a specific technical manual. To find out which one, read on.

The Machine Reference Manuals

The *Cortland Hardware Reference* and the *Cortland Firmware Reference* contain information about the machine itself. You don't need to read these manuals to be able to develop applications for the Cortland, but they will give you a better understanding of the machine's features. They will also provide the reasons why some of those features work the way they do.

The Toolbox Manuals

Like the Macintosh, the Cortland has a built-in toolbox that can be called by applications. The toolbox serves two purposes: it makes developing new applications easier, and it supports the desktop user interface.

When you first start using the toolbox, *Writing Cortland Programs* provides the recommendations and guidelines you need. It is not a complete course in programming for the Cortland; rather, it is a starting point. It explains the Cortland tools and describes an event-driven program. It includes a simple example of such a program that uses the Cortland tools, and demonstrates the way you use the Cortland Programmer's Workshop to develop the program.

For detailed specifications of the tool calls, you'll need both volumes making up the *Cortland ToolBox Reference*.

The Cortland Programming Languages

The Cortland does not restrict developers to a single programming language. Apple is currently providing an assembler and compilers for C and Pascal. Other compilers can be used with the workshop, provided that they observe the standards Apple has set up.

There is a separate reference manual for each programming language on the Cortland. The manuals for the languages Apple provides are the *Cortland Assembler Reference*, the *Cortland C Compiler Reference*, and the *Cortland Pascal Compiler Reference*.

The Programmer's Workshop Manual

The core of the development environment on the Cortland is the Cortland Programmer's Workshop, also called CPW. CPW is a set of programs that enable developers to create and debug application programs on the Cortland. The manual that describes CPW is the *Cortland Programmer's Workshop* manual. It includes information about the parts of the workshop that all developers will use, regardless which programming language they use: the shell, the editor, the linker, the debugger, and the utilities.

What About ProDOS?

ProDOS on the Cortland comes in two flavors: one for compatibility with the models of Apple II that use 8-bit CPUs, called ProDOS/8, and one that utilizes the full power of the Cortland, ProDOS/16. Those two versions of ProDOS are described in their own manuals, *ProDOS/8 Technical Reference* and *ProDOS/16 Technical Reference*.

All-Apple Manuals

In addition to the Cortland manuals mentioned above, there are two manuals that apply to all Apple computers. Those are *Human Interface Guidelines* and *Apple Numerics Manual*.

How to use this book

If you are planning on simply using the Cortland Tools that Apple provides, Chapters 1 and 2, along with the specifications for the individual tools and routines, will provide you with enough information. If you are planning on developing your own tools, you will need to read Appendix A.

What this manual contains

This manual contains the following chapters:

- About this manual tells you about this manual.
- Chapter 1, *Introducing the Tools*, defines the terms used in this manual and summarizes the capabilities of the Tools.
- Chapter 2, *Calling the Apple Tools*, describes how to call the tools from assembly language, C, and Pascal, describes how to pass parameters, and indicates how errors are signaled.
- The rest of the chapters describe the individual tools, one to a chapter. All of the routines available for the tool are included in its appropriate chapter.
- Appendix A provides the information needed to write your own tool set, and Appendix B summarizes all tool error codes for quick reference.

Visual Cues

(Not yet decided for Grand Design manuals.)

Other Reference Material You'll Need

(TBD)

Language Notation

(Still being hotly debated.)

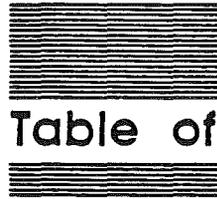


Table of Contents

Chapter 1 Introducing The Tools

What is a Tool?

What can the Tools do for me?

Are there any limitations?

What kinds of Apple Tools are there?

Integer Math Tools

Event Manager

Menu Manager

Miscellaneous Tools

QuickDraw II

SANE Tools

Sound Manager

Tool Locator

Window Manager

Chapter 2 Using the Apple Tools

Initializing Tools at application start-up

Calling the correct routine

Calling a routine from assembly language

Calling a routine from Pascal

Calling a routine from C

Passing Parameters

Return from the call

Flags and registers

Error handling

Chapter 3 Event Manager

Overview

Event Types

Mouse Events

Keyboard Events

Window Events

Other Events

Event Priority

Event Records

Event Code

Event Message

Modifier Flags

Event Masks

Using the Event Manager

Responding to mouse events

Responding to keyboard events

Responding to window events

Responding to other events

Posting and removing events

Other operations

Using alternative pointing devices

Installing device drivers

Removing device drivers

The journaling mechanism

EMBootInit

EMStartUp
EMShutDown
EMVersion
EMReset
EMActive
DoWindows
GetNextEvent
EventAvail
GetMouse
Button
StillDown
WaitMouseUp
PostEvent
FlushEvents
GetOSEvent
OSEventAvail
TickCount
GetDBTime
GetCaretTime
SetSwitch
SetEventMask
FakeMouse
Event Manager Error Codes

Chapter 4 Integer Math Tools

(The rest of the tool chapters will be carried out to the same level of detail as the Event Manager)

Chapter 5 Memory Manager

Chapter 6 Menu Manager

Chapter 7 Miscellaneous Tools

Chapter 8 QuickDraw II

Chapter 9 SANE

Chapter 10 Sound Manager

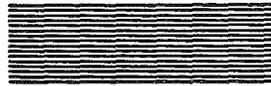
Chapter 11 Text Tools

Chapter 12 Tool Locator

Chapter 12 Window Manager

Appendix A Writing your own tool set

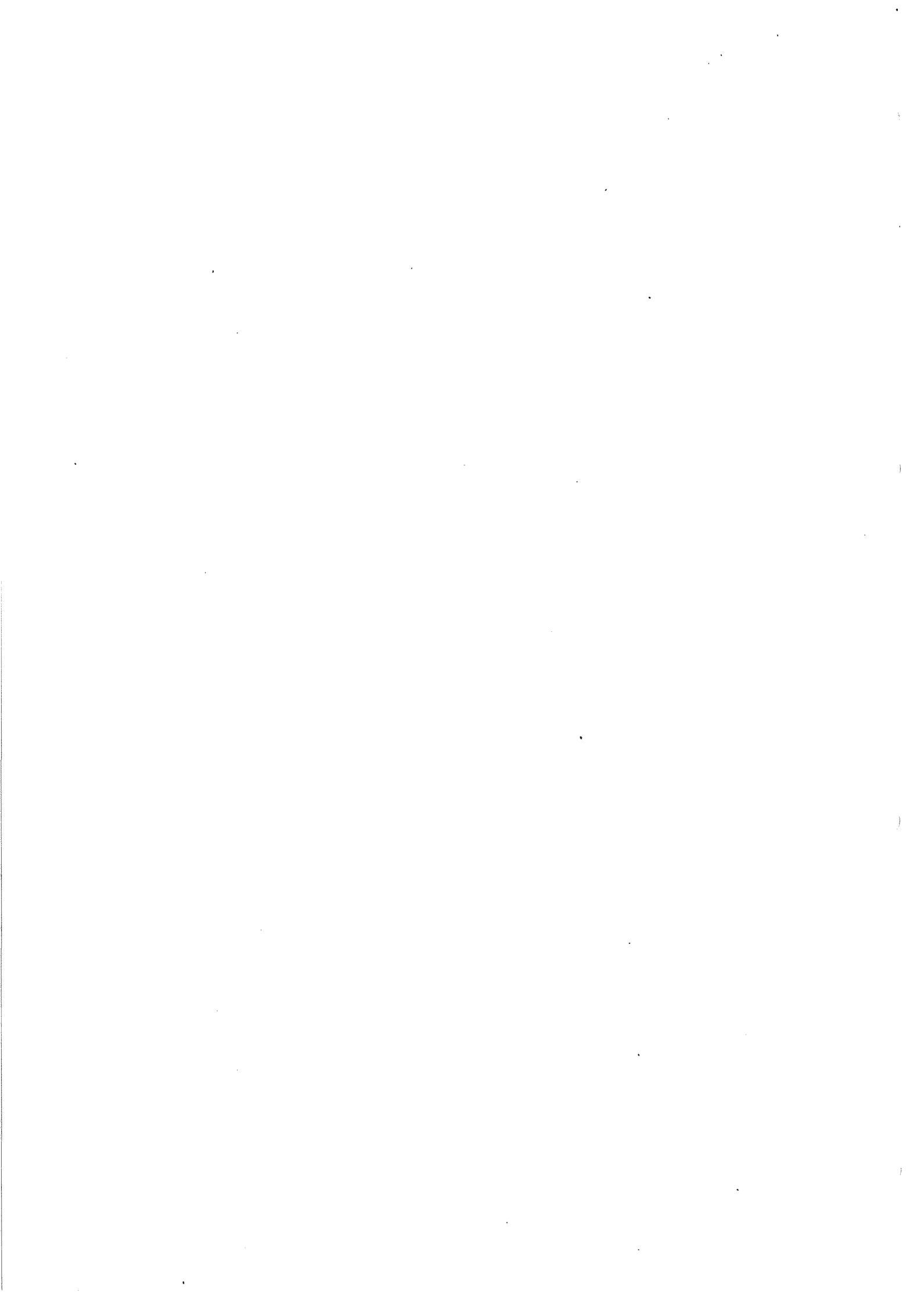
Appendix B Error Codes



Chapter 1



Introducing the Tools



What is a Tool?

A software "Tool" or "Tool Set", in the Cortland environment, is a collection of related routines (or functions) that provides one major capability. Each routine performs a fundamental operation and converts zero or more inputs to zero or more outputs and side effects. For example, the QuickDraw II Tool provides routines that handle graphics on the Cortland. Within that tool, PenSize and PenMode are functions that set the pen size and pen mode.

The tools, then, are routines that are always available to perform many common tasks. If you are familiar with Macintosh programming, this concept is similar to the Macintosh Toolbox. In the Cortland implementation, the concept is even more important. Many of the capabilities of the Cortland are easily accessed through the tools. For example, even the Memory and Event Managers are considered to be tools on the Cortland. ("Manager", by the way, is simply another name for a collection of routines. Some of the tool sets are called "xxx Tools", others are called "xyz Manager", with the names assigned merely by convention).

What Can the Tools Do For Me?

The tools provide powerful capabilities that allow an application to concentrate on its specific business rather than on the background work .

A number of the tools are included in ROM. This approach makes those tools available to all programs without using disk space. Additional tools are available in RAM. However, you don't need to keep track of where a particular function is or even if it is in ROM or RAM. A tool called the Tool Locator, which allows tools and applications to communicate, takes care of the necessary bookkeeping functions.

The Tool Locator fires up when the Cortland is turned on, and thereafter does its work behind the scenes. You won't even need to call the Tool Locator if you are simply using the Apple tools in your application. To use the tools in the simplest fashion, you don't need to know anything but the name for the tool and how to call it from the appropriate programming language. (Calling information is in Chapter 2.)

The tools thus provide their capabilities at a minimum cost; their bookkeeping functions are almost automatic, the interface to them is simple, and the applications you write will not be rendered obsolete by any future changes to the hardware.

The Tool Locator is also flexible enough to allow you to extend the scope of the tools by writing your own, and powerful enough to keep track of both the Apple tools and your tools. You can write and install your own tools if you wish, and still have the Apple tools available when needed.

The Cortland tools are independent of the operating system being used. They are thus available for any Cortland application, whether the application is running under ProDOS, Pascal, or another operating system.

Are There Any Limitations?

There is at least one important point to consider when you are planning to call a Cortland tool from your application: the tools are designed to run in "full native" mode, rather than in Apple II emulation mode. In full native mode, the e, m, and x bits registers are all set to 0, which provides native mode, a 16-bit accumulator, and 16-bit index registers. Almost all of the tools require this mode, and simply will not work if the machine is in any other state. The limited exceptions to this rule are documented under the individual calls described in later chapters in this manual.

(** Are there any other factors that should be mentioned? Should I warn them away from hybrid applications and the tools? **)

What Kinds of Apple Tools Are There?

In this section, we simply list the tool sets, the categories within the tool sets, and a brief summary of some their capabilities. The listing does not contain definitions of the calls; for a summary of the routines available for a given tool, look at the first page of the chapter describing the tool. For an individual routine, look it up in the index and under its appropriate tool.

We recognize that a summary such as this can seem like teasing; in fact, that's some of the point! The Cortland toolbox is so large that we wish to introduce you to their entire range and encourage you to use as many of them as possible.

Every tool set or manager has a class of routines known as "Standard Housekeeping Routines". These routines allow the tool set to be dealt with as a Cortland tool set. Included among the routines are boot initialization and application startup calls, an application shutdown call, a reset call, and a call which returns the version number of the particular tool and status. Whether or not a particular type of these standard calls is used varies from tool set to tool set; however, all of the routines must be present in each tool set.

Integer Math Tools

These routines support multiplication and division of several types of numbers, and also convert numbers from one type to another. The types of numbers dealt with are as follows:

- Integers, which are single word signed integers
- Long integers, which are two-word signed integers
- Fixed, which are two-word signed values with 16 bits of fraction
- Frac, which are two-word signed values with 30 bits of fraction

Math Routines

These routines support multiplication and division of integer, long integer, Fixed, and Frac numbers.

Conversion Routines

These routines convert between a binary value and an ASCII character string representing that value. The binary value can be either a 2-byte integer or a 4-byte integer. The character string can be in either hexadecimal or decimal format.

The routines allow you to:

- Convert integers to hex, long, or decimal format ASCII strings.
- Convert longs to decimal format ASCII strings.
- Convert hex ASCII strings to integer or long.
- Convert decimal format ASCII strings to integers or longs.

Event Manager

Toolbox Event Manager routines

These routines check events to see if they are of interest to the application. If the events are of interest, and the Desk Manager doesn't want them, the routines return with the event.

Mouse reading routines

These routines provide the ability to read the status of the mouse.

Posting and Removing Events

These routines allow you to place or remove events into the event queue.

Accessing Events Routines:

These routines check events to see if they are of interest to the application. If the events are of interest, the routines return with the event.

Miscellaneous Event Manager Routines:

These routines allow you to:

- Check the number of ticks(sixtieths of a second) since the system last started up.
- Return suggested maximum difference of ticks which determines a double mouse-click..
- Return the number of ticks between blinks of the caret marking the insertion point.

Menu Manager

(Writer's note: this is a RAM tool and is still changing shape. Summary will be in next draft.)

Miscellaneous Tools

The miscellaneous tools are a collection of various routines. Their capabilities are summarized below.

Battery RAM Routines

These routines allow you to:

- Write or read data to or from the Battery RAM.
- Write or read data to or from a specified Battery RAM parameter.

Clock Routines

These routines allow you to set or get the current time.

Vector Initialization Routines

These routines allow you to set or get the vector address for a specified interrupt manager or handler.

HeartBeat Routines

These routines allow you to:

- Install or delete a specified task into or out of the HeartBeat Interrupt Service queue.
- Remove all tasks from the Heartbeat Interrupt Service queue.

System Death Manager

This routine allows you to control the System Death Manager.

Get Address Tool

This routine allows you to determine the address of a parameter referenced by the firmware.

Mouse Routines

These routines allow you to:

- Initialize, set, position, home, and read the values for the mouse.
- Set and get the clamp values for the mouse.
- Return the interrupt status for the mouse.

ID Tag Manager Routines

These routines allow you to:

- Create and delete ID tags used for memory management.
- Return the status of a given ID tag.

Interrupt Control

This routine allows you to enable or disable certain interrupt sources.

Firmware Entry

This routine allows you to use some AppleII emulation mode entry points.

Tick Counter

This routine returns with the current value of the tick counter.

Packing and Munging Tools

These routines allow you to:

- Pack bytes into, and unpack bytes from, a special format which uses less storage space.
- Manipulate bytes in a string of bytes.

Interrupt Enable State

This routine returns with certain hardware interrupt enable states.

Absolute Clamp Routines

These routines set and get the current values of the absolute device clamps.

QuickDraw II

QuickDraw II is the graphics handler for the Cortland. Since many of the other Cortland Tools depend on QuickDraw II (the Window Manager, for example), the QuickDraw calls will be used by virtually every application.

There are several categories of QuickDraw II calls, as discussed in the following sections.

Global environment routines

These routines set up the graphics environment for QuickDraw II and the other tools. Included are calls which:

- Specify ("Set") or return ("Get") the settings for the scan line bytes, the color tables, and the entries in the color tables.
- Set or get the setting for the system font.
- Set or get the setting for the maximum width or size of buffers dealing with text.
- Clear the screen.
- Turn Super Hi-Res Graphics mode on or off.

GrafPort routines

These routines set up the GrafPort for QuickDraw II and the other tools. Included are calls which:

- Open, initialize, and close a GrafPort.
- Set or get the current Grafport or the GrafPort's location.
- Set, get, change the size, adjust the origin, or move the GrafPort's PortRect.
- Set, get, or change the current clip regions.
- Hide or show the pen, and set or get the current values for the state, size, mode, pattern; and mask for the pen.
- Set or get the background pattern.
- Move the current pen location to a point or a relative distance.
- Set or get the current font, the font flags, the font globals, and other font information.
- Set or get the text face and mode.
- Set or get the space and char extra fields.
- Set or get the foreground and background colors.
- Set or get the ClipRgn, VisRgn, and VisHandle.
- Set or get the GrafProcs record.

Drawing routines

These calls allow you to:

- Draw a line from the current pen position to either a specified point or a relative distance.

- Frame, paint, erase, invert, or fill rectangles, regions, polygons, ovals, round rectangles, and arcs.

Pixel transfer routines

These routines allow you to scroll or shift a region of pixels.

Text drawing and measuring routines

These routines allow you to:

- Draw a character, text, string, or c-string.
- Get the width of a character, text, string, or c-string.
- Fill a rectangle with a character, text, string, or c-string.

Calculations with rectangles

These routines allow you to:

- Set the dimensions of a rectangle.
- Offset or inset a rectangle.
- Calculate the intersection of two rectangles and place the intersection in a third rectangle.
- Calculate the union of two rectangles and place the union in a third rectangle.
- Determine whether a point is in a particular rectangle, or copy points to the upper left and lower right of a rectangle.
- Determine whether two rectangles are equal.
- Determine whether a rectangle is empty.

Calculations with points

These routines allow you to:

- Add two points together, or subtract a point from another point.
- Set a point to specified values.
- Determine whether two points are equal.
- Convert a point from local to global coordinates, and visa versa.

Calculations with regions

These routines allow you to:

- Create a new region, or dispose of a region.
- Copy contents from one region to another.
- Empty or overwrite a region.
- Open or close a temporary region.
- Offset or inset a region.
- Calculate the union of two regions and place the union in a third region.
- Calculate the intersection of two regions and place the intersection in a third region.
- Calculate the difference between two regions and place the difference in a third region.
- Calculate the difference between the union and the intersection of two regions and place the result in a third region.
- Determine whether a point is in a particular region.
- Determine whether a rectangle intersects a particular region.
- Determine whether two regions are equal.
- Determine whether a region is empty.

Calculations with polygons

These routines allow you to open, close, dispose of, or offset a polygon.

Mapping and scaling utilities

These routines allow you to:

- Map points, rectangles, and regions from a source to a destination.
- Scale points from a source to a destination.

Miscellaneous utilities

These routines allow you:

- Return a pseudorandom number.
- Set a seed value for a random number generator.
- Get the values for a specified pixel.

Customizing QuickDraw operations

These routines are similar to their Macintosh counterparts. They allow you to:

Set up a standard Proc record.

Draw standard text, lines, rectangles, round rectangles, ovals, arcs, polygons, regions, and pixels.

Make standard comments for pictures.

Do standard text measuring.

Do standard storage and retrieval from the text record.

Tell QuickDraw not to use scan line interrupts.

Get the address of a screen table and two conversion tables.

Cursor-handling routines

These routines allow you:

- Initialize the cursor.
- Set or get the current settings for the cursor.
- Show or hide the cursor.
- Obscure the cursor.

Sane Tools

The ROM Tools for the Cortland will provide all of the functions found in the Standard Apple Numeric Environment (SANE). The SANE Tools can be called by using the normal call mechanism. For more information regarding the capabilities of SANE, refer to the *Apple Numerics Manual*.

Sound Manager

The Sound Manager controls sound generation for the Cortland, particularly for the Digital Oscillator Chip (DOC). There are two groups of calls in the manager; the tool calls and some low-level routines designed for fast access.

Sound Tool Calls

These routines allow you to:

- Write and read a specified number of bytes from and to DOC RAM.
- Set and get the volume for a sound generator, or change the system volume.
- Start or stop the sound for a particular generator.
- Return the status of a specified generator, or the status of all generators.
- Set up the entry points for the system and user sound interrupt handler.
- Return the current Free Form synthesizer sound-playing status.
- Return the jump table address for the low-level routines

Low-Level Sound Routines

These routines, designed for quick access, allow you to:

- Write and read any register within the DOC.
- Write and read a specified Ensoniq RAM location.
- Write and read the next DOC or RAM location.

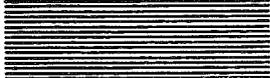
Tool Locator

The Tool Locator provides the magic that allows the Cortland Tools to function. You'll only need to use it if you are writing your own Tool Set to supplement the Apple Cortland Tools.

The functions provided by this tool set get and set the Tool Set Pointers, Function Pointers, and Work Area Pointers all of the Tools need. For the other tools, the Tool Locator provides these functions automatically.

Window Manager

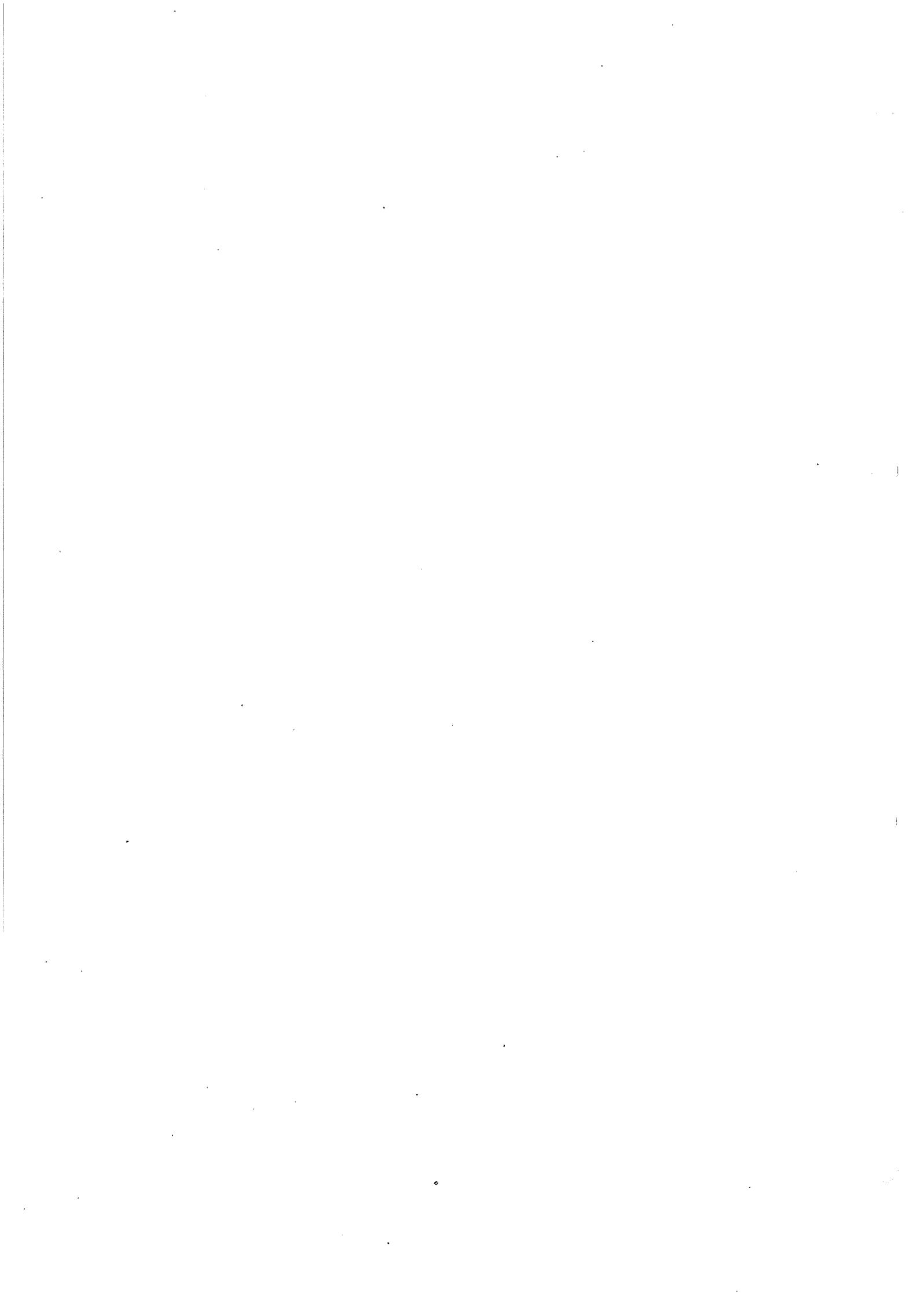
(Writer's note: this is a RAM tool and is still changing shape. Summary and details will be in next draft.)



Chapter 2



Using the Apple Tools



Initializing Tools At Application Start-Up

There will be a simple mechanism for asking for the correct tools when the application starts. However, design of the mechanism is not yet complete.

Calling the Correct Routine

The Apple Tools are available from 65816 assembly language, ??? Pascal, and Cortland Workshop C. The general rules for accessing the Tools are outlined in the following sections.

Calling a Routine From Assembly Language

We have provided macros in order to make calling a routine as simple as possible. You make an assembly-language call as follows:

1. If the function has any output, push room for it on the stack.
2. Push the inputs in the specified order listed.
3. Invoke the appropriate macro by entering the name of the routine.
4. Pull the output, if any, from the top of the stack.

The input and output parameters for each call are provided in the descriptions of each individual call in this manual and in the *Cortland Toolbox Reference: Volume II* (not yet available).

Calling a Routine From Pascal

(**Writer's note: the information in this section is pure conjecture, based on the information available for C, since Pascal remains in limbo.**)

The interface libraries which allow the Cortland tools to be accessed from the Pascal programming language are included in Cortland Workshop Pascal. Those libraries contain the function and procedure definitions for the tools. The steps to use a particular routine are as follows:

1. Make the routine accessible by providing a USES statement which includes the appropriate file (for example, ??? for a Quickdraw II call). The file will provide the function and procedure declarations.
2. Invoke the call by entering its name and supplying the correct parameters.

The names of the USES files and the parameters for each routine are described in the individual routine descriptions in this manual and in the *Cortland Toolbox Reference: Volume II* (not yet available).

Calling a Routine From C

The interface libraries which allow the Cortland tools to be accessed from the C programming language are included in Cortland Workshop C. Those libraries contain the function definitions for the tools. The steps to use a particular routine are as follows:

1. Make the routine accessible by using an #include statement which includes the appropriate file (for example, quickdraw.h for a Quickdraw II call). The included file will provide the function declarations.
2. Invoke the call by entering its name and supplying the correct parameters.

The names of the #include files and the parameters for each routine are described in the individual routine descriptions in this manual and in the *Cortland Toolbox Reference: Volume II* (not yet available). (**Writer's note: the libraries and function calls were not available soon enough to include in this draft.**)

Passing Parameters

Most input and output parameters to the tool calls are passed on the stack, with some occasional exceptions. The parameters and parameter-passing method are defined by each routine. Usually, the parameters are passed on the stack, with the routine pulling input parameters off and leaving any output parameters on the stack for the calling program to handle. The method and parameters for each routine is described under the routine in the chapters which describe specific Tool Sets.

Return From the Call

Upon completion of the call, the routine returns control directly back to the application. The following sections describe the state of the flags and registers upon return from a tool call and the way that errors are returned from the call.

Flags and Registers

The state of all flags and registers upon return from a tool call is summarized in the following table:

Table X-X: Flags and Registers on Return From a Call

N flag	As set by routine
V	As set by routine
m	Unchanged (must be 0)
x	Unchanged (must be 0)
D	Set to 0
I	Unchanged
Z	As set by routine
C	As set by routine, or error flag (see next section)
E	Unchanged (must be 0)
A register	As set by routine or A≠0 successful call, A≠0 error code (see next section)
X	As set by routine
Y	As set by routine
S	Parameters have been removed from stack
D	Unchanged
P	See list of flags above
DB	Unchanged
PB	Unchanged
PC	Address following call

Note that "unchanged" means that the value is the same as it was just before the function call.

Error Handling

Some tools can return errors on some routines. If they do, the convention is that the carry flag (C flag) is set to 1 if an error occurred, and the A register contains the error code. The error code has the following format:

High Byte	Low Byte
Tool Set Number	Message Number

With this method, an error can be properly identified even if it occurs during one tool set's call, but doesn't actually show up until a call from another tool set. For example, using this method, a QuickDraw II call can pass on an error message from the Memory Manager.

The error codes for an individual Tool Set are listed at the back of its chapter, and all error codes are summarized in Appendix B.

Error codes\$0001-\$000F are reserved for use by the function dispatcher. Remaining error codes are defined by each tool set..

