Technote 1167

NewWorld Architecture

By Paul Resch
Revised by Wayne Flansburg
Apple Worldwide Developer Technical Support

The Mac ROM is "different" starting with the iMac. Come along and find out what's new.

CONTENTS

Introduction

The NewWorld Architecture

What's Different?

NewWorld Components

Boot Process Overview

Name Registry Overview

Background Information

Downloadables

This Technote describes changes made to the Macintosh ROM since the introduction of the iMac.

The Macintosh ROM, sometimes called the ToolBox ROM, has been updated. The ToolBox (including the OS) has been removed from the ROM; the ROM physical size, Macintosh memory map, and boot sequence have also been changed.

This Technote describes the changes to the new Apple ROM called NewWorld, which will be the ROM used on all future Macintoshes. This Note is directed at device developers who have devices such as PCI, USB, and FireWire (especially device types that could participate in the boot sequence).

Introduction

The NewWorld Architecture is the basis for Mac OS startup and ToolBox functionality for all Macintosh CPUs beginning with iMac. This document is designed to help developers understand the organization of the NewWorld Architecture and some ways to use it to best advantage.

This document describes how the NewWorld Architecture works from an organizational and execution flow standpoint, and describes differences from older architectures. It briefly covers the "Old World" ROM organization as background, then explains the NewWorld Architecture and execution flow.

Familiarity with the traditional Macintosh ROM structure is useful when reading this document.

While the focus of this document is on Mac OS, the Boot ROM and "bootinfo" components of the NewWorld Architecture are designed to be operating-system independent. Furthermore, the mechanisms behind the engineering techniques used for Mac OS can be applied to other operating systems.

Back to top

NewWorld Architecture

ROM is a computer's permanent read-only memory. Historically, the Macintosh ROM (also called the Macintosh ToolBox ROM) has been structured as one monolithic ROM, containing both low-level and high-level code. That is, it contained the routines needed by the computer at power-up time (hardware knowledge, initialization, diagnostics, drivers and such), as well as quite a bit of higher-level Mac OS ToolBox code.

While a computer needs to have a ROM with hardware-specific code in order to boot, the higher-level code was also included in the Macintosh ROM because the Mac ROM had its genesis in the original 128K Macintosh computer back in 1983-84. In those days, ROM was cheaper than RAM, and the available disk space (which was floppy based) was at a premium.

The NewWorld Architecture breaks the hardware-specific and higher-level system software into two logically distinct pieces. Under this model, one piece holds most of the hardware-specific components needed to boot, while the other contains boot-time ToolBox routines and hardware-specific components that are common to many Macintosh computers.

Hardware-specific code still exists in firmware (ROM) in order to handle the computer's start-up activities. This code fits into one ROM called the Boot ROM. The Boot ROM has the hardware-specific code and description of the hardware needed to start up the computer, as well as to boot an OS and provide common hardware access services the OS might require. One part of the Boot ROM contains Open Firmware. This Open Firmware implementation is significantly improved over versions of Open Firmware found on older PCI-based Macintosh computers. In particular, the device tree and Open Firmware drivers are much more complete.

Higher-level ToolBox software is no longer in the Boot ROM. This software has been moved to a "ToolBox ROM Image". This image behaves like the old ToolBox ROM, but is loaded from mass storage. As before, this ToolBox ROM Image can still be augmented by what traditionally was contained in Enablers, the System file, and extensions. The ToolBox ROM Image exists as a file, and the ToolBox ROM Image is inserted into the memory map as if it were a ROM (that is, it is write- protected in the memory map).

Back to top

What's Different?

From a user and developer point of view, the NewWorld Architecture is implemented to be as compatible with previous systems as possible. "It's just a Mac!" is a goal of the architecture. This section briefly describes the differences.

Memory (RAM)

A big change that the NewWorld Architecture introduces is that RAM is not mapped one-to-one, as it has been for previous PCI-based Macs. This means that well-behaved software calls `LogicalToPhysical` and/or `PrepareMemoryForIO` functions. Software that assumes the logical and physical addresses are the same—even when Virtual Memory is not on—will fail.

Hardware Addresses

Hardware components, including the PCI bridge and the interrupt controller, are not located at the same addresses as on previous PCI-based Macintosh computers. The Mac OS Name Registry provides the

addresses, and it should be used to determine all addresses, which may change again.

Name Registry

The Mac OS Name Registry contains the Open Firmware Device Tree, which is more complete than pre-NewWorld implementations. The Name Registry functionality is unchanged by the NewWorld Architecture except to add new mechanisms that provide communication with Open Firmware. The "configuration variables" used by Open Firmware during boot are modifiable from Mac OS using existing Name Registry API calls. Properties saved in NVRAM by Mac OS software are available to Open Firmware drivers as well as Mac OS drivers. See the New Name Registry Functionality section for more details.

gestaltMachine Type

All NewWorld-based CPUs have the same `gestaltMachineType` (406 decimal). Any software that uses `gestaltMachineType` to verify that this is a valid CPU on which to execute needs to be changed to add checks for the existence of the hardware it requires by using the Name Registry.

Interrupt Handling

Although the API calls related to interrupt handling have not changed, the code that handles interrupts is very different under the NewWorld Architecture. The new interrupt code allows for dynamic handling of the interrupt "wiring." The interrupt latency has been reduced to such an extent as to make it negligible.

Fewer Resources in ROM

Many resources in the old ToolBox ROM exist in the System Folder as well, often as replacements that fix or enhance those in the ToolBox ROM, but other times because the resources have not yet been removed from the ToolBox ROM. Many of the resources that are not needed early in the boot sequence are no longer in the ToolBox ROM Image, and more will be removed as schedule permits.

ROM-in-RAM

The NewWorld Architecture puts the ToolBox ROM Image in RAM, and marks it read-only. Although the image is 4 megabytes in size, not all of those 4 megabytes are in use. The portion that is not used is returned to Mac OS for use as part of RAM. At the time this document was written, less than 3 megabytes of the 4 megabyte ToolBox ROM Image are in use, allowing more than 1 megabyte to be returned to Mac OS as available RAM.

NVRAM and PRAM

Instead of using hard-coded offsets to locations in NVRAM for Mac OS PRAM and other non-volatile information, NewWorld breaks NVRAM into variable-sized partitions that are used by Mac OS, Open Firmware, and any other client. The partitioning scheme is part of the CHRP specification. PRAM resides in the Mac OS partition, and API calls to read and modify PRAM refer to offsets with that partition. Properties saved in NVRAM are saved in an Open Firmware configuration variable. See the New Name Registry Functionality section for more details.

USB

The USB Manager is in a NewWorld ToolBox ROM Image, along with class drivers for USB hubs, keyboards, and mice.

ADB

Macintosh computers no longer always include ADB hardware. For compatibility, the ADB Manager still functions, treating USB keyboards as a variant of an ADB keyboard. This added compatibility does not allow all ADB devices to work as if ADB hardware still exists, even if a USB-ADB conversion device is attached to a USB connector.

Floppy

NewWorld supports Macintosh CPUs with or without floppy drives. Apple does not support floppy-based copy protection on CPUs that do not have Apple floppy drives. Providing functionality similar to the Apple floppy driver is the responsibility of the developer of software for mass storage devices that can read and/or write floppy diskettes. The most expedient way to do so is to take over the Apple floppy driver slot in the drive queue.

Video Drivers

To present a seamless transition between the Open Firmware user interface and the user interface used by Mac OS, the NewWorld Architecture provides a mechanism for communication of the display settings (mode, resolution, and so forth) between the Open Firmware video driver and the OS video driver. See the New Name Registry Functionality section for more details.

Back to top

NewWorld Components

The two main areas involved in the NewWorld Architecture are the Boot ROM and the "bootinfo" file, along with supporting changes in Mac OS System Software. Before implementation of the NewWorld Architecture, the Mac OS ToolBox ROM contained all of the hardware-specific initialization code and the Mac OS-specific start-up and ToolBox functions. The NewWorld Architecture breaks up that monolithic Mac OS ToolBox ROM into several pieces that are located in two places.

The Boot ROM is a physical part of the specific implementation of the Macintosh CPU, containing:

- POST (Power-on Self Test), start-up code without Mac OS-specific code
- diagnostics
- boot beep, error beep
- Open Firmware

Mac OS drivers (`'ndrv's` and `'nlib's`) for motherboard devices needed at boot time. The "bootinfo" file is kept in the System Folder of the startup volume containing:

- Mac OS-specific Open Firmware code and required "bootinfo" components
- Open Firmware-specific Mac OS code ("Trampoline" code)
- a Mac OS ToolBox ROM Image
- and other Mac OS software

Other operating systems also use the Boot ROM and a "bootinfo" file, but the contents of the "bootinfo" file are specific to the operating system. The contents of the "bootinfo" file listed above are for Mac OS only. See the Background Information section for more information.

The "bootinfo" file exists on the boot device and has a localized name. Identification information that leads to the file's path is stored in the Open Firmware configuration variables in NVRAM. A search algorithm for a usable "bootinfo" file parallels the search mechanism across SCSI, ATA, and other interfaces used in the older Mac OS startup implementations. By default, the file is located by using the "blessed folder" `dirID` in the Master Directory Block and then searching for a file with a "file type" of

`'tbxi'`. Searching by file type is done to allow localization of the file. If non-localized, the name of the "bootinfo" file is "Mac OS ROM"; this name may change.

Some versions of the Mac OS "bootinfo" file contain what has been traditionally part of an **enabler**. This is only to reduce the number of files in the System Folder, and Open Firmware does not use the enabler components in any way.

Back to top

Boot Process Overview

The following list is a high level view of the execution path take when a NewWorld-based computer boots Mac OS:

1. The POST code runs (preliminary diagnostics, boot beep, initialization, and setup). This is similar to code in a Mac OS ToolBox ROM, but it is different in that it does not contain OS-specific code.
2. Open Firmware initializes and begins execution, including building the Device Tree.
3. Open Firmware loads the "bootinfo" file, based on defaults and NVRAM settings.
4. Open Firmware executes the "Forth" script in the "bootinfo" file, which contains information about the rest of the file and instructions to read both the Trampoline code and the ToolBox ROM Image and place them into a temporary place in memory.
5. The "Forth" script transfers control to the Trampoline code, which functions as the transition between Open Firmware and the beginning of the Mac OS execution.
6. The Trampoline code gathers information about the system from Open Firmware, creates data structures based on this information, terminates Open Firmware, and rearranges the contents of memory to an interim location in physical memory space.
7. The Trampoline code transfers control to the ToolBox ROM Image initialization code.

Startup and the Startup Disk Control Panel

The boot sequence, up to loading and execution of the Mac OS ROM image, is controlled by Open Firmware. To provide a user experience like previous Macintoshes, Open Firmware now supports searching for possible boot devices and user overrides of boot devices using keyboard input such as Command-Shift-Option-Delete or "C" to force booting from a CD-ROM.

When the user selects a startup device in the Startup Disk control panel, Startup Disk no longer sets a value in Mac OS PRAM. Instead it generates an Open Firmware path to the device and saves that path in NVRAM as Open Firmware's "boot-device" configuration variable. Open Firmware tries the device specified by "boot-device" first. If this device is unavailable or the user has overridden this with keyboard input, Open Firmware scans other devices looking for bootable drives. Once Open Firmware selects a device, it sets the "bootpath" property in the "chosen" node to the path to that device. The "bootpath" property is what the Mac OS ROM subsequently uses to locate and load Mac OS from disk.

In order for a device to be bootable, Open Firmware needs methods for accessing the drive. For built-in devices, such as SCSI and ATA, these methods are supplied by Open Firmware. For plug-in cards, the PCI configuration ROM on each card must supply these methods for the card as specified in the PCI Open Firmware binding.

Back to top

Name Registry Overview

Greater dependence on Open Firmware requires additional functionality in Mac OS. The Name Registry continues to provide a database that includes the device tree. NewWorld extends the Name Registry API to provide a mechanisms to update Open Firmware configuration variables and to communication between Mac OS and Open Firmware drivers.

Modifying Open Firmware configuration Variables from the Mac OS

Name Registry provides a mechanism to save and restore device tree properties in NVRAM. This mechanism has been augmented. Using the same mechanism, software can modify Open Firmware configuration variables. The mechanism uses API calls to create and modify device tree properties and the `RegistryPropertySetMod` API call, using the `kRegPropertyValueIsSavedToNVRAM` modifier bit. If these calls are made to the properties in the "device-tree:options" node, the corresponding Open Firmware configuration variables in NVRAM are modified.

Communicating Between Mac OS Drivers and Open Firmware FCode Drivers

In addition to providing access to Open Firmware configuration variables, the same Name Registry mechanism that saves and restores device tree properties in NVRAM has new functionality that allows communication between a Mac OS driver and its corresponding Open Firmware driver. The driver sets a property in its device tree node, setting the `kRegPropertyValueIsSavedToNVRAM` modifier bit. This property is saved in a special Open Firmware configuration variable that is used by Open Firmware during boot to restore properties into the device tree. Since the properties exist when an Open Firmware driver is opened, such a property can be a message from the Mac OS driver to the Open Firmware driver. The Mac OS driver will also find the property during its initialization. The new mechanism extends the size limitations for properties saved in NVRAM to 8 bytes for the name and 32 bytes for the data.

Back to top

Background Information

bootinfo

The PowerPC Microprocessor Common Hardware Reference Platform (CHRP) System binding to: IEEE Standard 1275-1994 Standard for Boot (Initialization, configuration) Firmware document is the basis for the "bootinfo" file format and use. A "bootinfo" file contains Open Firmware script, a description, information for individual operating systems, icons, along with other information. A "bootinfo" file can be extended to contain non-Open Firmware information, including, for Mac OS, the ToolBox ROM Image.

CHRP

Common Hardware Reference Platform—A specification jointly developed by Apple, Motorola, and IBM to provide a way that non-Apple system developers could build Macintosh-compatible computers. Although the specification was published, Apple did not agree to license Mac OS to non-Apple system developers. See PowerPC Microprocessor Common Hardware Reference Platform: A System Architecture.

Open Firmware

Open Firmware provides boot mechanisms and a description of the system hardware, in the form of a "device tree". The IEEE Standard 1275-1994 Standard for Boot (Initialization, configuration) Firmware: Core Requirements and Practices document, along with associated bindings, provides the specification for Open Firmware. The current bindings are available at:

IEEE 1275 at Sun

or at Apple's mirror site at:

IEEE 1275 at Apple

Back to top

Downloadables

 Acrobat version of this Note (49K).

Back to top

Acknowledgments

Thanks to Mark Baumwell for co authoring the original draft. Thanks to Dave Radcliffe for the update to the Startup Disk Control Panel to set the Open Firmware boot-device Configuration Variable.

To contact us, please use the Contact Us page.
Updated: 17-May-1999

Technotes | Contents
Previous Technote | Next Technote