
PCI
SIG™

PCI Firmware Specification Rev 3.0 Overview

Tony Pierce

Firmware Workgroup Chair

Microsoft Corporation

Agenda

- History, New Purpose and Scope
- Structure Overview
- Reporting MMConfig
- Specification Timeline
- PCI Option ROM Management
  - ✓ Historical Issues
  - ✓ PCI Option ROM Management in the PCI 3.0 Firmware specification

History, New Purpose and Scope

- History
  - ✓ PCI BIOS Specification Revision 2.1
    - Released August 26, 1994
  - ✓ Systems over the last several years use a multitude of firmware technologies for PCI, e.g. ACPI
- New Purpose and Scope
  - ✓ Update 2.1 content
  - ✓ Centralize and Advance Option Rom Content
  - ✓ Incorporate appropriate references and descriptions for other firmware specifications
  - ✓ Provide all updates needed for PCI-X 2.0 and PCI Express

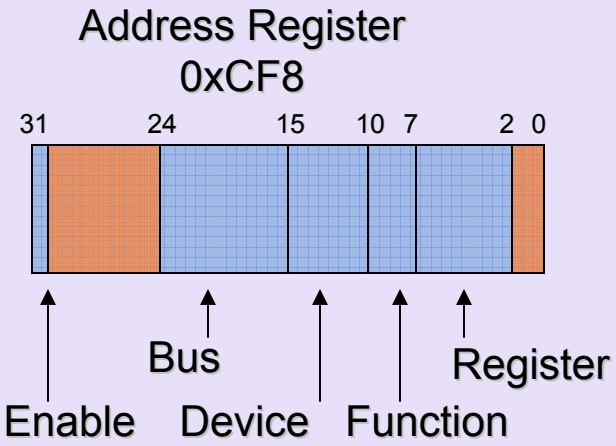
Structure Overview

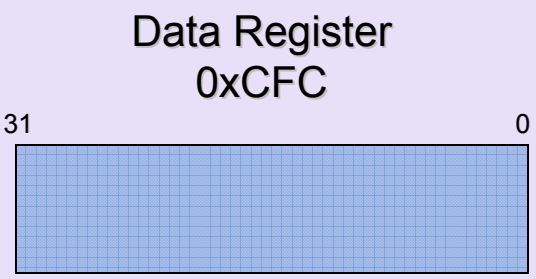
- Chapter 1- Introduction
- Chapter 2-Traditional PCI BIOS
  - ✓ Updated Rev 2.1 Content
- Chapter 3- EFI PCI Services
- Chapter 4- PCI Services in ACPI
- Chapter 5- PCI Expansion ROMs
  - ✓ Consolidated Content
- Chapter 6- PCI Services Specific to DIG64-compliant Systems

Reporting MMConfig

- Background
- New Model
- Declaring Configuration Space Region
- Multi-Root Machines and Segments

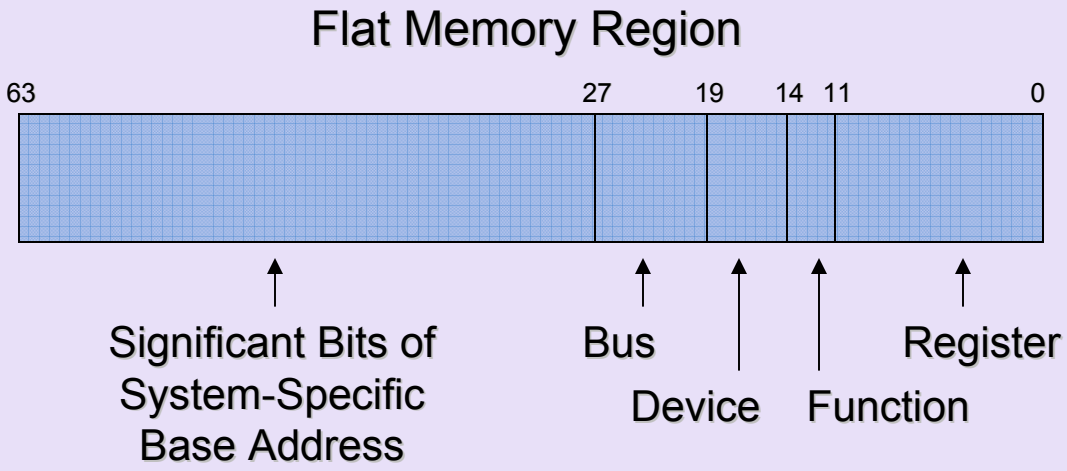
Legacy Access Model





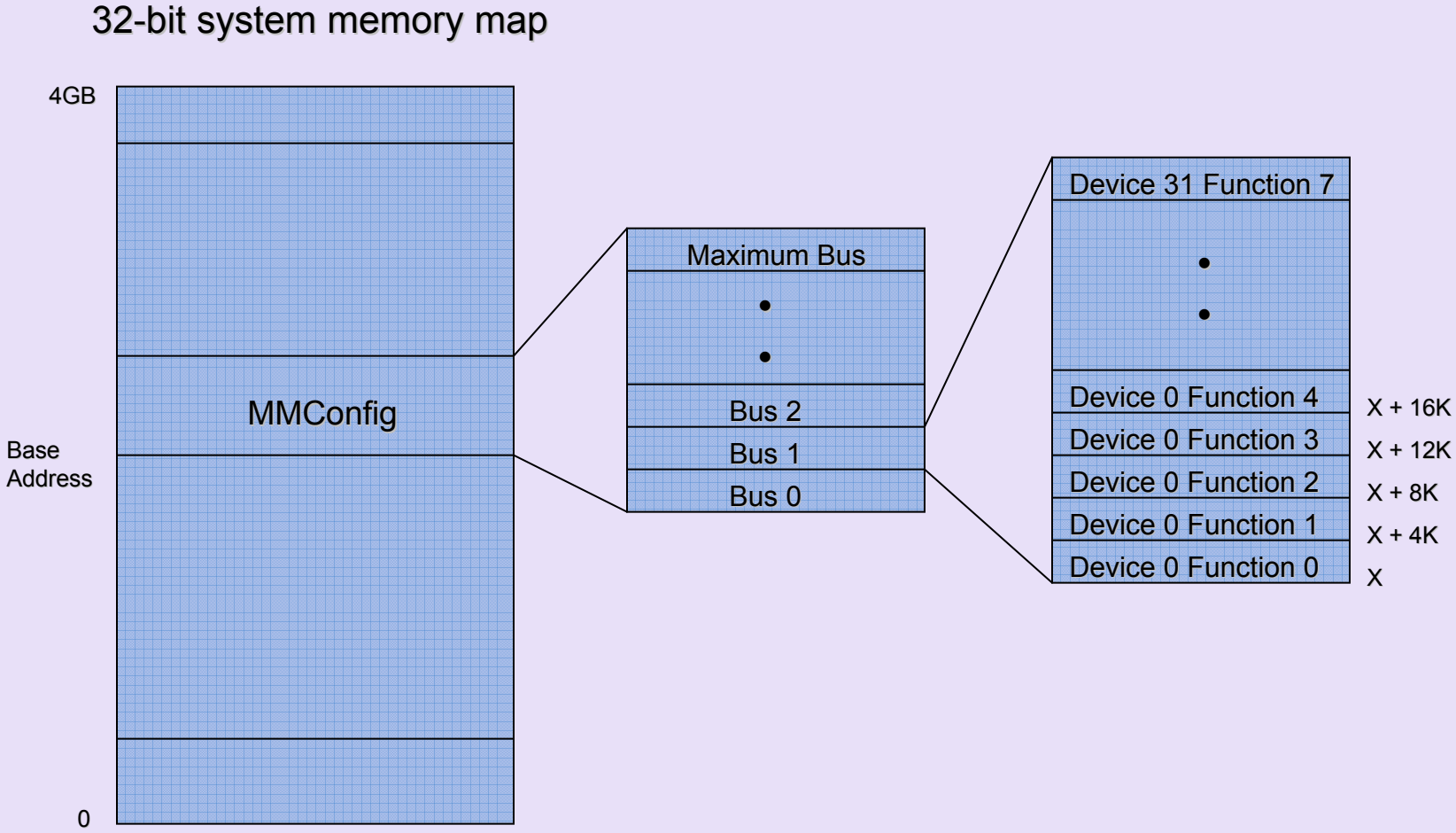
- I/O Mapped
- Addresses identical across machines
- Only supports 256 bytes/device
- Only supports 256 buses
- IA-64 is done differently

MMConfig Model



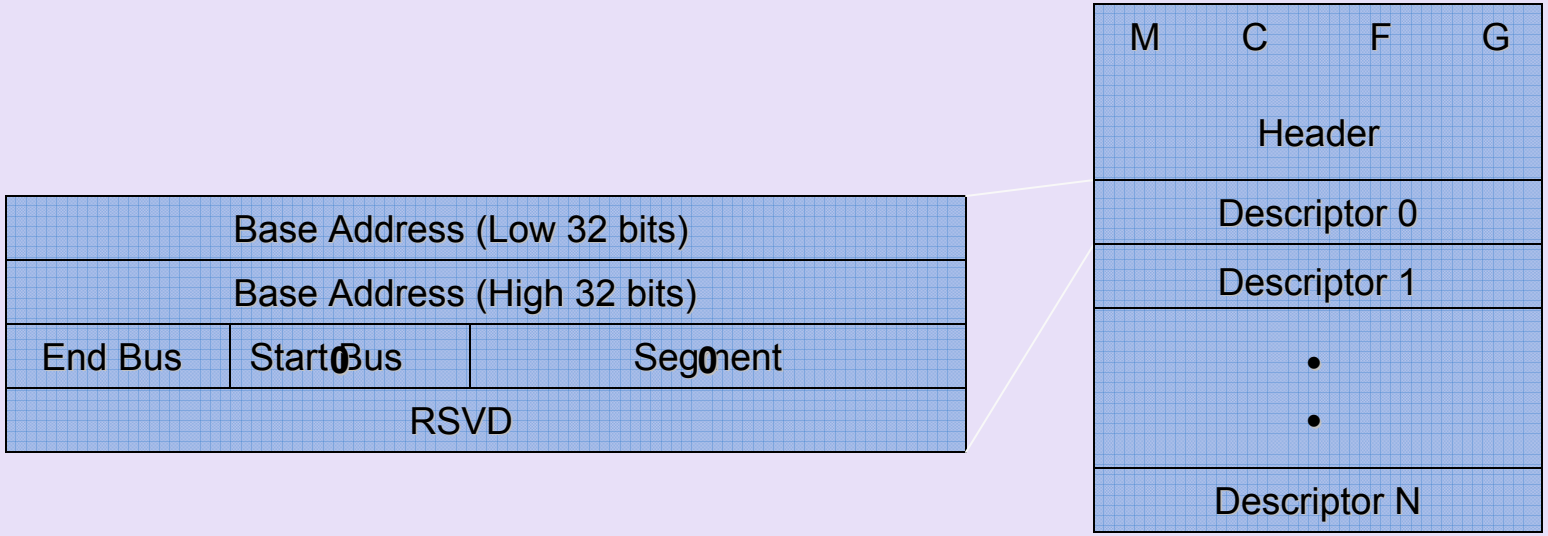
- Memory Mapped
- Requires up to 28 bits of memory
  - ✓ 256MB
- Supports 4K bytes/device
  - ✓ Each device is on its own 4K memory page
- Supports 256 buses per base address

MMConfig and the System Memory Map



Reporting Range through MCFG

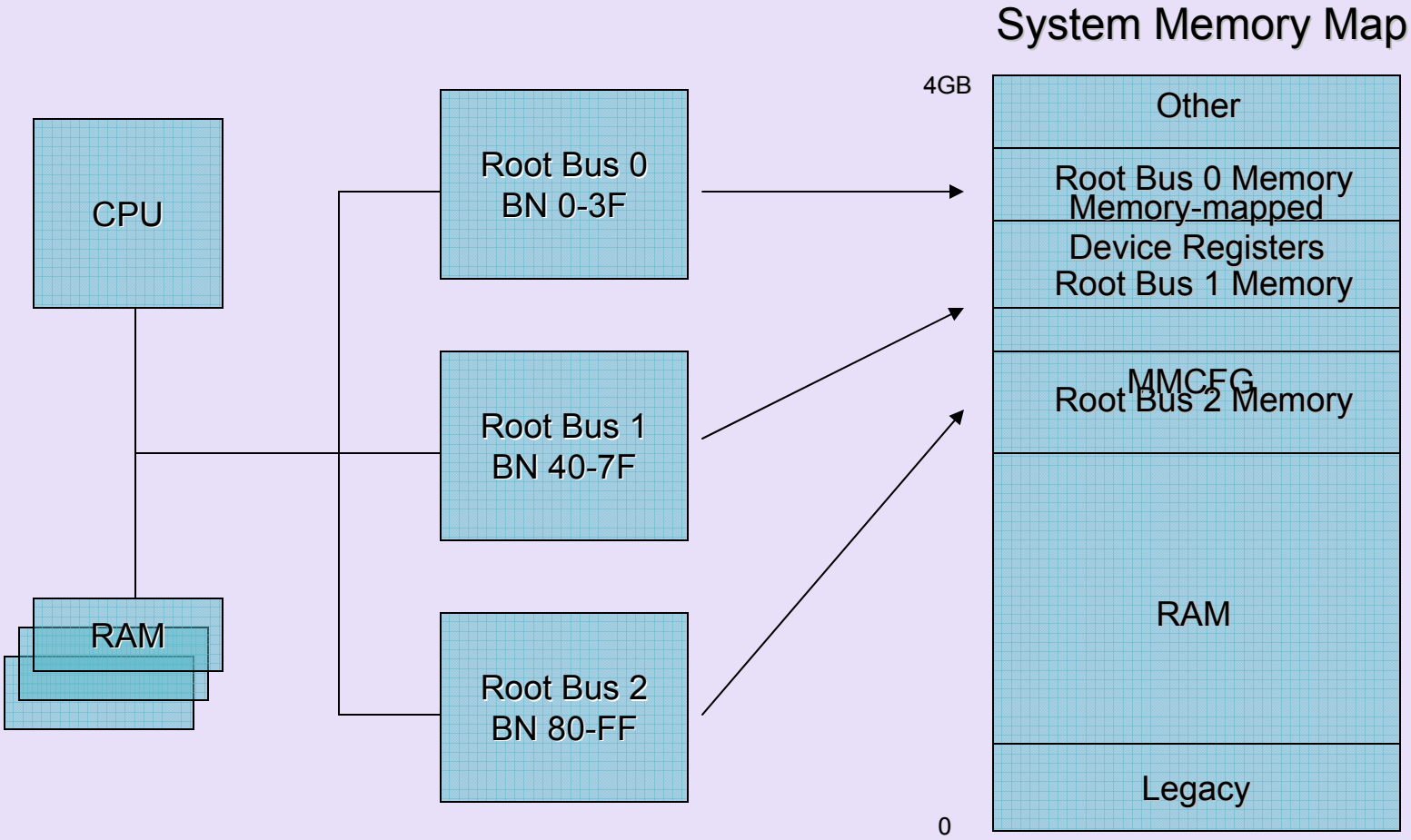
- ACPI table defined in PCI Firmware Spec v3.0
- Memory range described here is reserved
  - ✓ 1MB per bus
- Simple case
  - ✓ 1 segment
  - ✓ Start Bus will be 0
  - ✓ End Bus may or may not be 0xFF



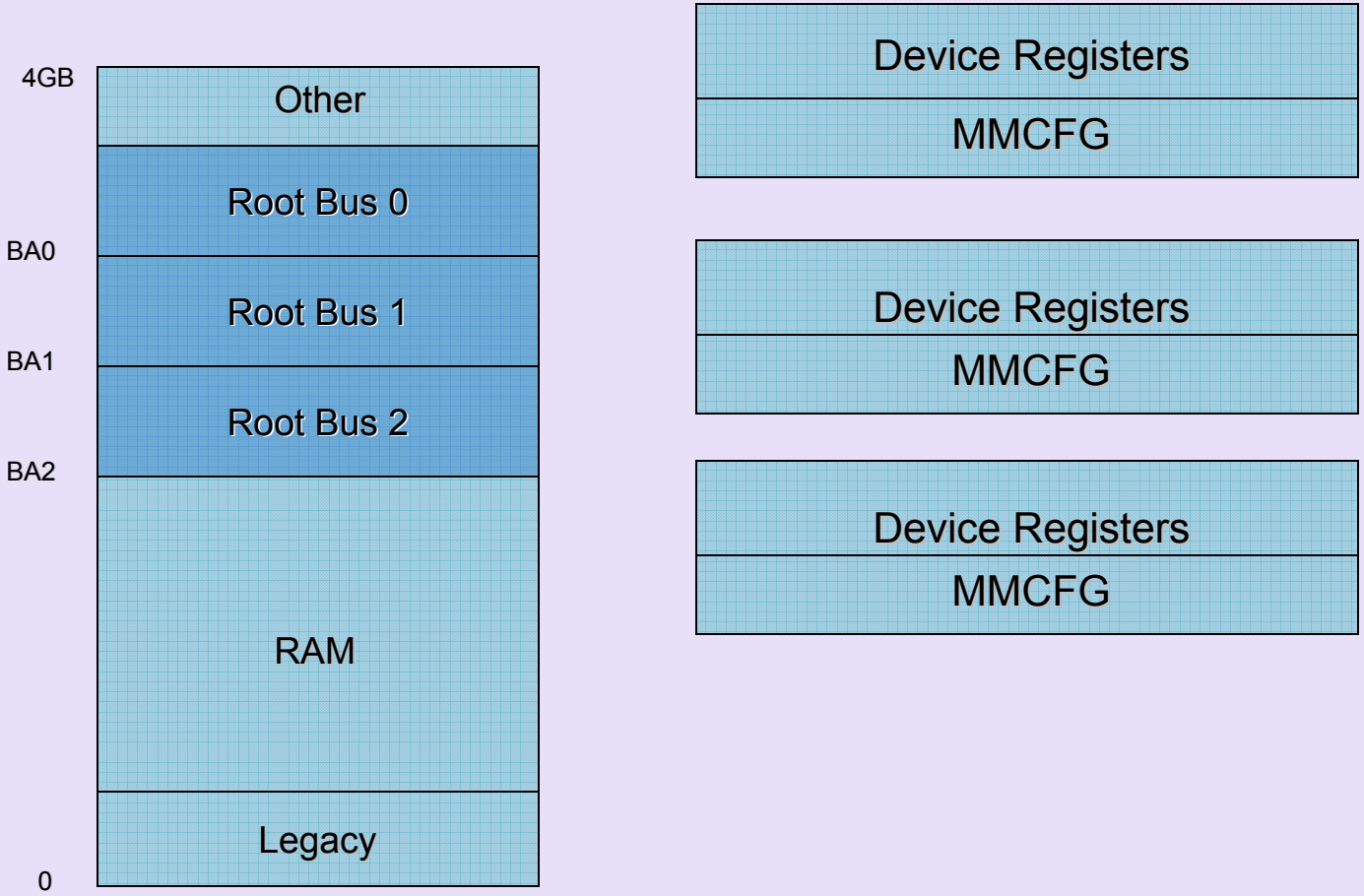
Reserving MMConfig Region

- Existing Operating Systems won't understand MCFG table
  - ✓ Backwards-compatible range reservation must be used
- For ACPI OS's Report range in ACPI "Motherboard Resources"
  - ✓ _CRS of PNP0C02 node
  - ✓ PNP0C02 must be at _SB scope
  - ✓ Range must be marked as consumed
- Do not include range in _CRS of PCI root bus
  - ✓ If included, OS will assume that this range can be allocated to devices
- E820 table/EFI memory map
  - ✓ May Not necessary to describe MMConfig here

Multi-Root Machines And MMConfig



Multiple Base Addresses



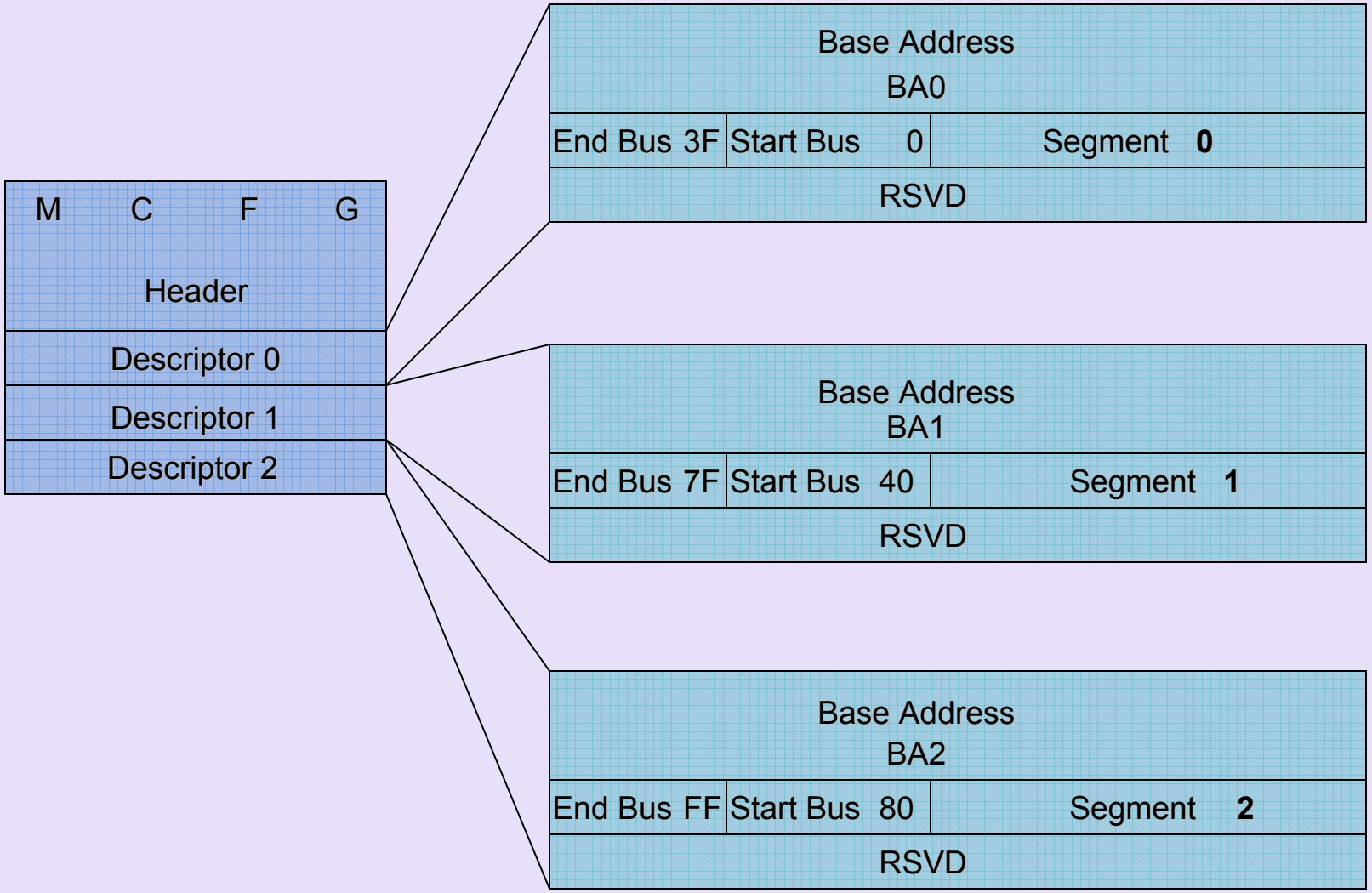
Multiple Base Addresses Means Segments

- MCFG table only supports one base address per segment
- Segment background
  - ✓ A way to get different bus number domains
    - E.g., two PCI-PCI bridges could have the same bus number
  - ✓ Requires a way to differentiate between segments when issuing config cycles
    - This is supported by the SAL, but wasn't by CFC/CF8
    - It is supported by memory-mapped config – each base address corresponds to a segment
- A new usage
  - ✓ Originally designed for systems where 256 buses weren't enough
  - ✓ These multi-root systems aren't bus-number constrained, just need multiple base addresses

Segment Transition

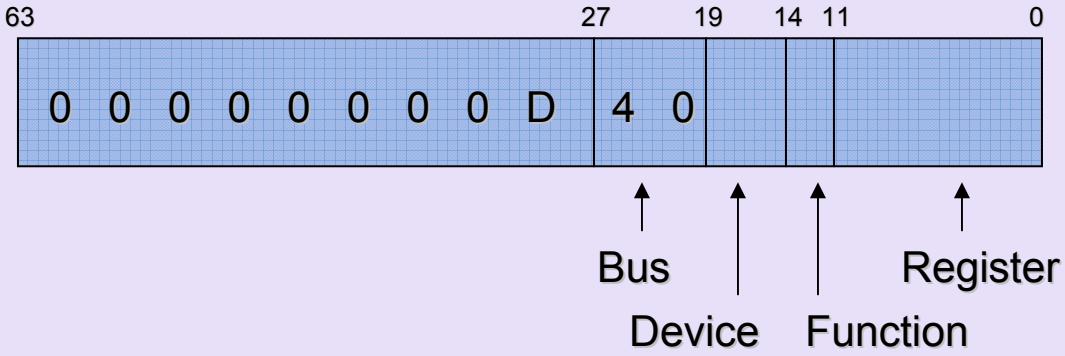
- Hybrid solution
  - ✓ Partition bus numbers among roots
  - ✓ Give each root a unique segment number
  - ✓ If segment information is ignored and CFC/CF8 is used, bus numbers don't conflict
  - ✓ If MMConfig is used, segment information can be used
- Simple solution
  - ✓ MCFG table is simple

Updated MCFG Table



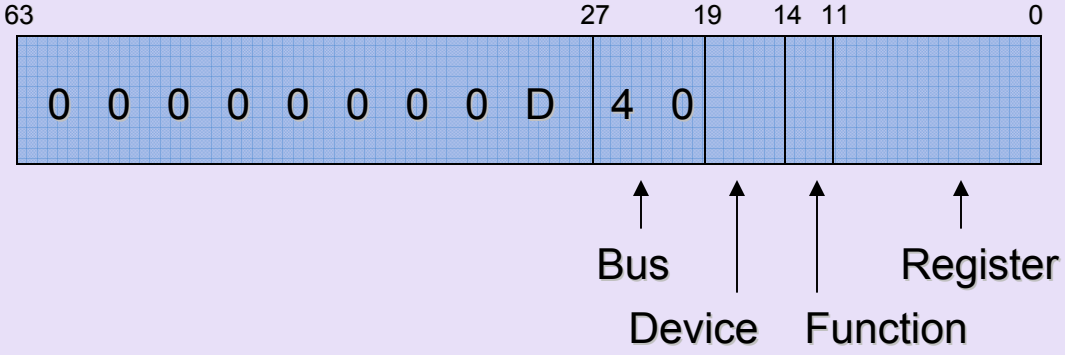
Example Multi-Segment Access

- Root Bus 1
  - ✓ Bus Numbers from MCFG – 40-7F
  - ✓ Base Address from MCFG – 0xD0000000
- Bus Number field is still the bus number to be accessed
  - ✓ NOT an offset



Reserving Multi-Segment Memory

- Root Bus 1
  - ✓ Bus Numbers from MCFG – 40-7F
  - ✓ Base Address from MCFG – 0xD0000000
- What memory gets reserved in PNP0C02?
  - ✓ First memory address is 0xD4000000
  - ✓ Last memory address is 0xD7FFFFFF
- Reserve 1MB / bus, starting at the first address that can be accessed
- If Start Bus is not 0, “Base Address” is NOT the base of the reserved range!



Specification Timeline

- Draft 0.7 Released this month
- Draft 0.9 Released in 3Q
  - ✓ Per new spec development process this starts 60day member IP review cycle
- Final 1.0 release in 4Q

PCI Option ROM Management

Trevor Western

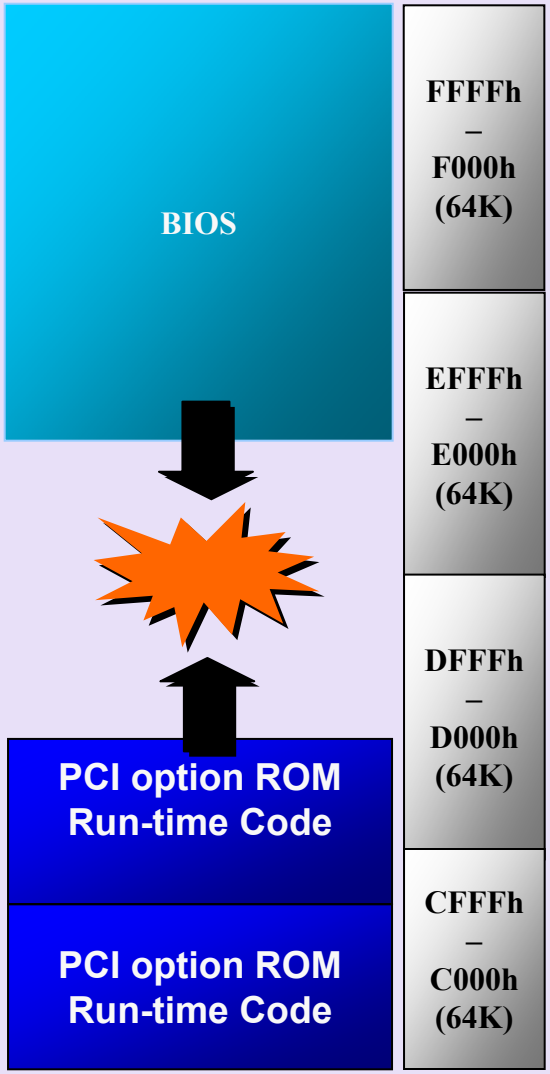
Phoenix Technologies Ltd

Historical issues

- Historical issues with PCI Option ROMs:
  - ✓ PCI 2.1 Option ROM specification was inadequate:
    - Did not fully describe BIOS POST behavior.
    - Did not fully describe PCI Option ROM behavior.
  - ✓ PCI Option ROMs were limited to one Device ID.
    - BIOS would match PCI Device to the Option ROM vendor ID and Device ID information. One-to-one match.

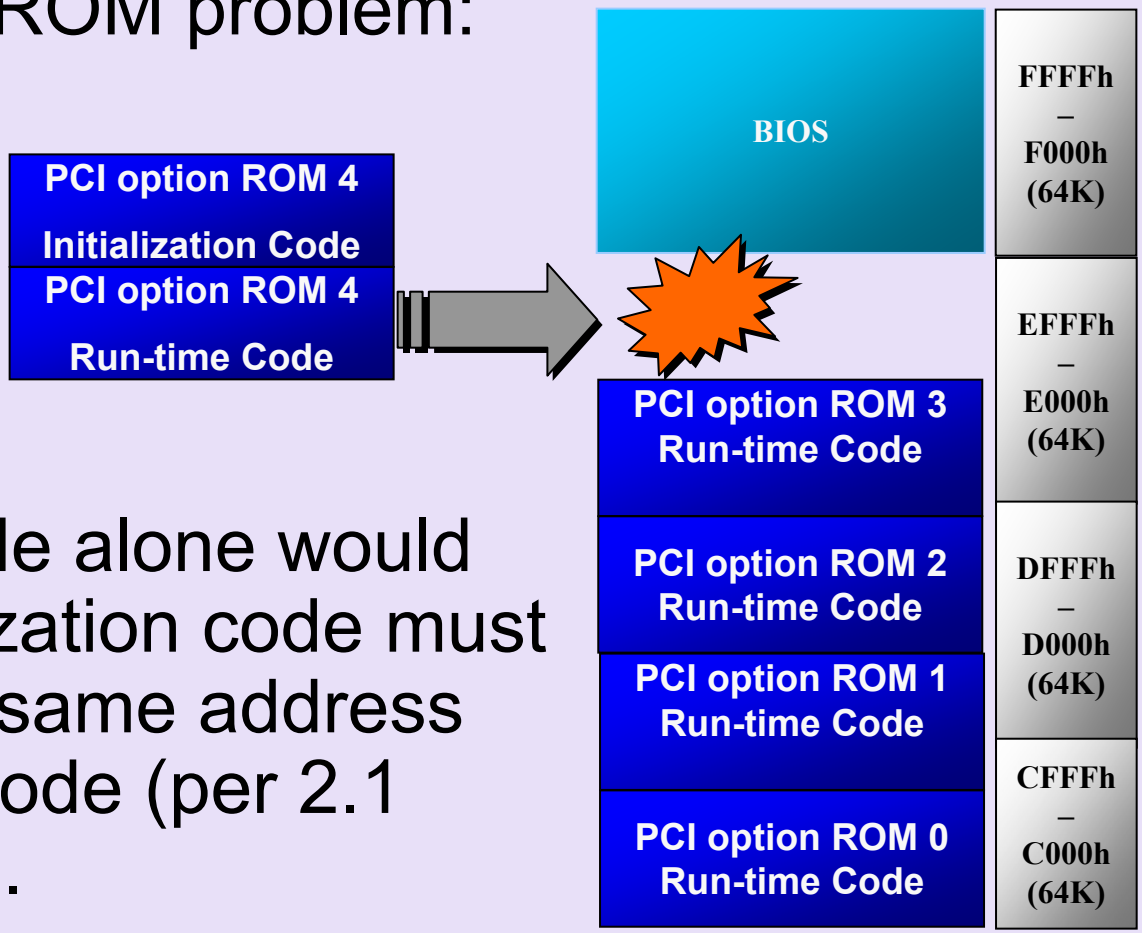
Historical issues

- BIOS can only place Option ROMs in a specialized memory region.
  - ✓ Limited space;
  - ✓ however Option ROMs are getting bigger,
  - ✓ and there's more of them.
  - ✓ BIOS is growing too.
- Limited to the C000h to FFFFh memory region.



Historical issues

- Final Option ROM problem:



- Run-time code alone would fit, but Initialization code must be placed at same address as run-time code (per 2.1 specification).

Historical issues

- Historical issues with PCI Option ROMs:
  - ✓ No legal way to obtain persistent memory blocks.
    - POST Memory Manager (PMM) only ran during POST.
    - PMM Memory was no longer assigned after OS ran.
    - Option ROM writers were forced to “steal” permanent memory.
  - ✓ Video Option ROMs must reside at C000h.
    - Drivers, apps, diagnostics have built in assumptions that if it’s C000h it must be video. Hold over from ISA 20+ years ago.

Historical issues

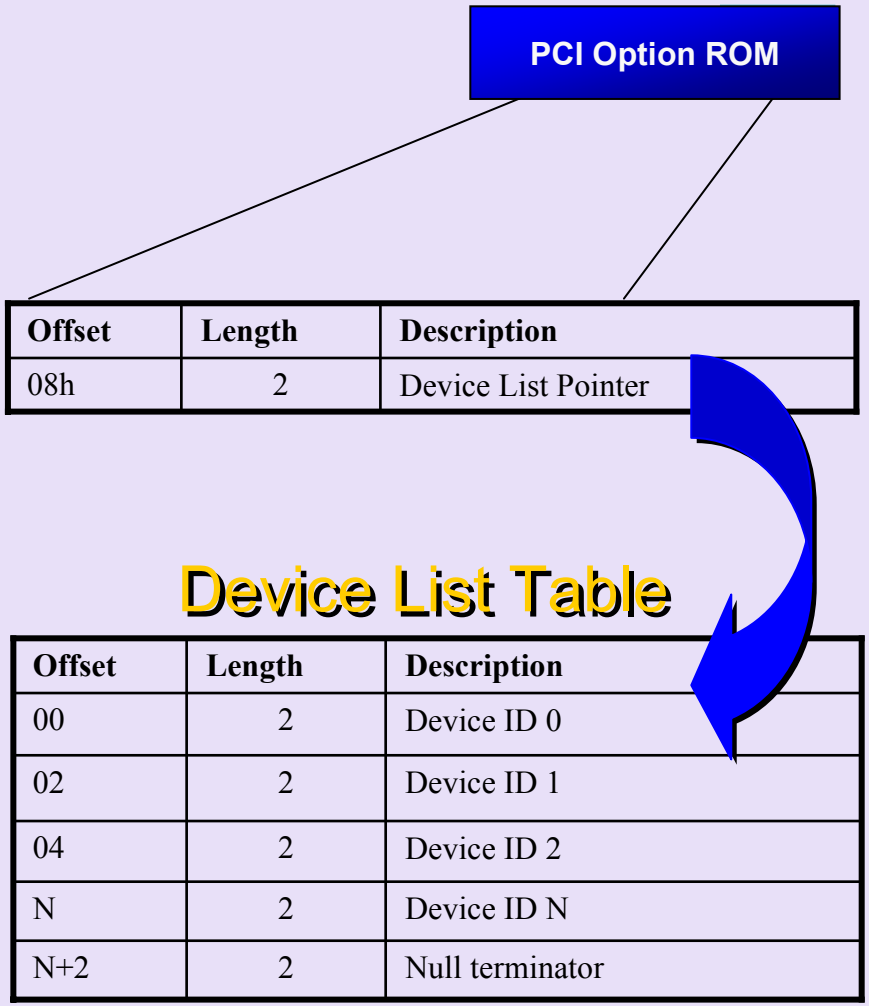
- Historical issues with PCI Option ROMs:
  - ✓ Option ROMs may run a user-interactive configuration utility.
    - Forces BIOS POST to wait for a special keystroke.
    - Configuration utilities with GUIs fail under Console Redirection.
  - ✓ BIOS cannot determine run-time size of Option ROM.
    - PCI Option ROMs can shrink to just run-time code after initialization. But how big?
    - Run-time code may fit in space remaining, but no way to be sure.
    - Option ROM not initialized because it may be too big.

PCI Option ROM Management in the PCI 3.0 Firmware specification

- PCI 3.0 Firmware specification solves these historical issues:
  - ✓ One Option ROM can be used for multiple devices.
  - ✓ Option ROMs can obtain permanently assigned memory for run-time use.
  - ✓ Option ROM placement range is expanded to A000h through FFFFh. (An extra 128K)
  - ✓ Video ROMs are can placed anywhere, not just C000h.
  - ✓ More efficient initialization of large Option ROMs.
  - ✓ Configuration Utilities are better managed.

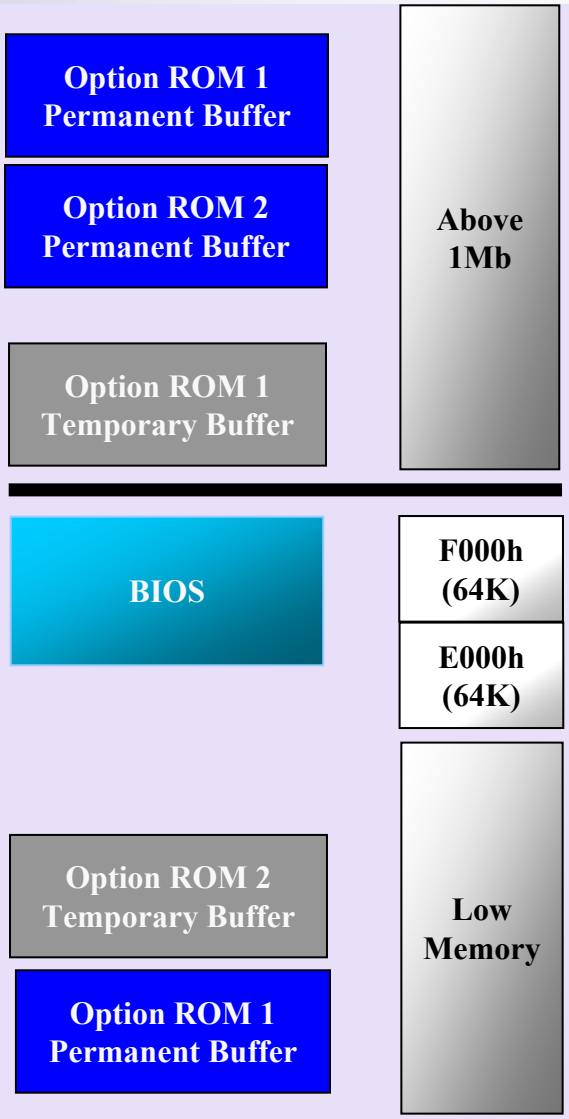
PCI Option ROM Management in the PCI 3.0 Firmware specification

- One PCI 3.0 Option ROM can initialize any number of PCI devices.
  - ✓ Device List Pointer added to PCI header.
  - ✓ Device List is a table of any number of Device IDs that can be associated with one Option ROM.
  - ✓ Reduces the number of PCI Option ROMs supplied by vendors.
- One-to-Many matching.



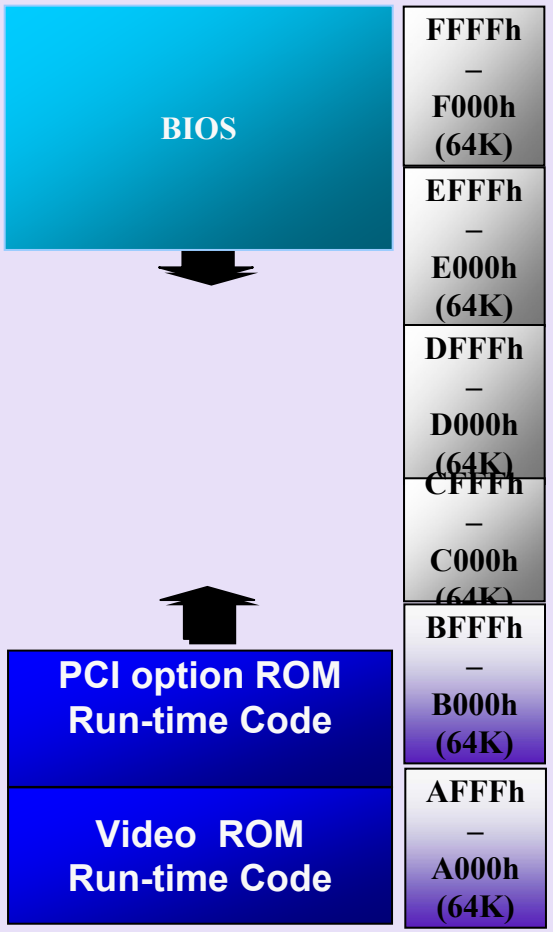
PCI Option ROM Management in the PCI 3.0 Firmware specification

- POST Memory Manager (PMM) definition expanded to include “permanent” memory.
  - ✓ An OS that understands Interrupt 15h E820h (“Get Memory Map”) will respect PMM memory.
  - ✓ An OS that understands ACPI “Get Memory Map” will respect the PMM memory.
  - ✓ BIOS must update the memory map with the new permanent memory allocations from PMM.
  - ✓ Limited to 64Kb above 1Mb, and 40Kb below 1Mb, per Option ROM.



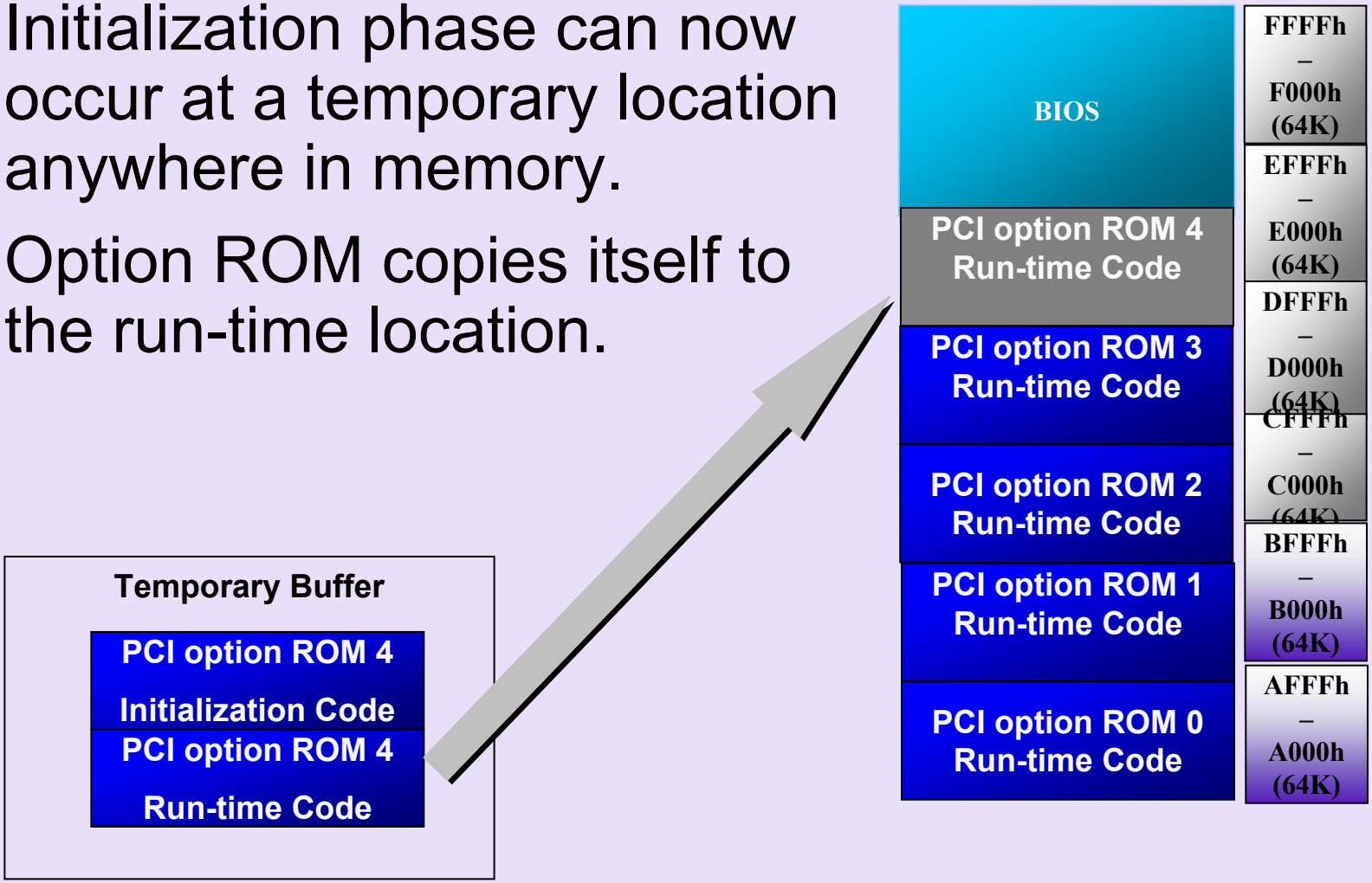
PCI Option ROM Management in the PCI 3.0 Firmware specification

- Placement region expanded to A000 – FFFFh. (Adds 128k more space).
- Video can go anywhere.
- **Caution:**
  - ✓ Some applications, diagnostics, and OSes may not expect video to be anywhere other than C000h.
  - ✓ Hardware may not be able to write protect A000h-BFFFh range.
- Works well in closed systems, with custom apps, OS, etc.



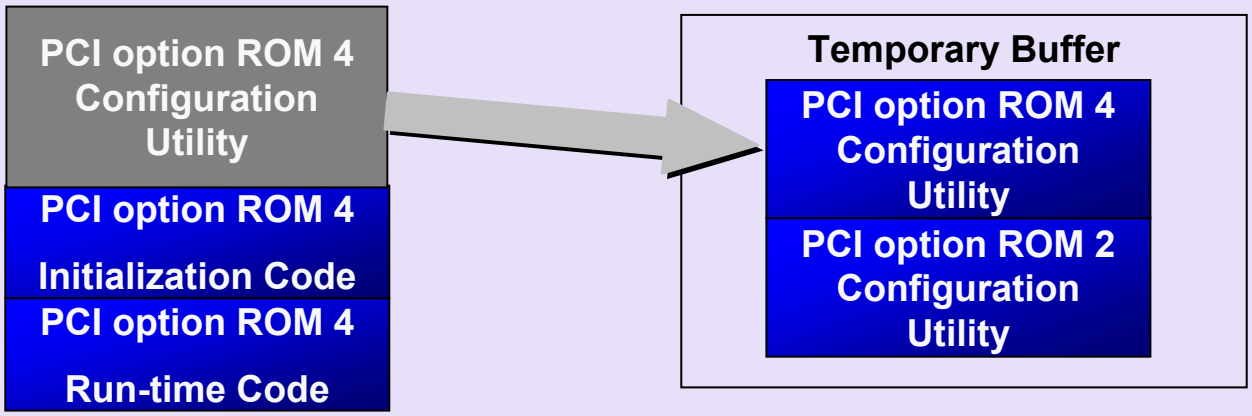
PCI Option ROM Management in the PCI 3.0 Firmware specification

- Initialization phase can now occur at a temporary location anywhere in memory.
- Option ROM copies itself to the run-time location.



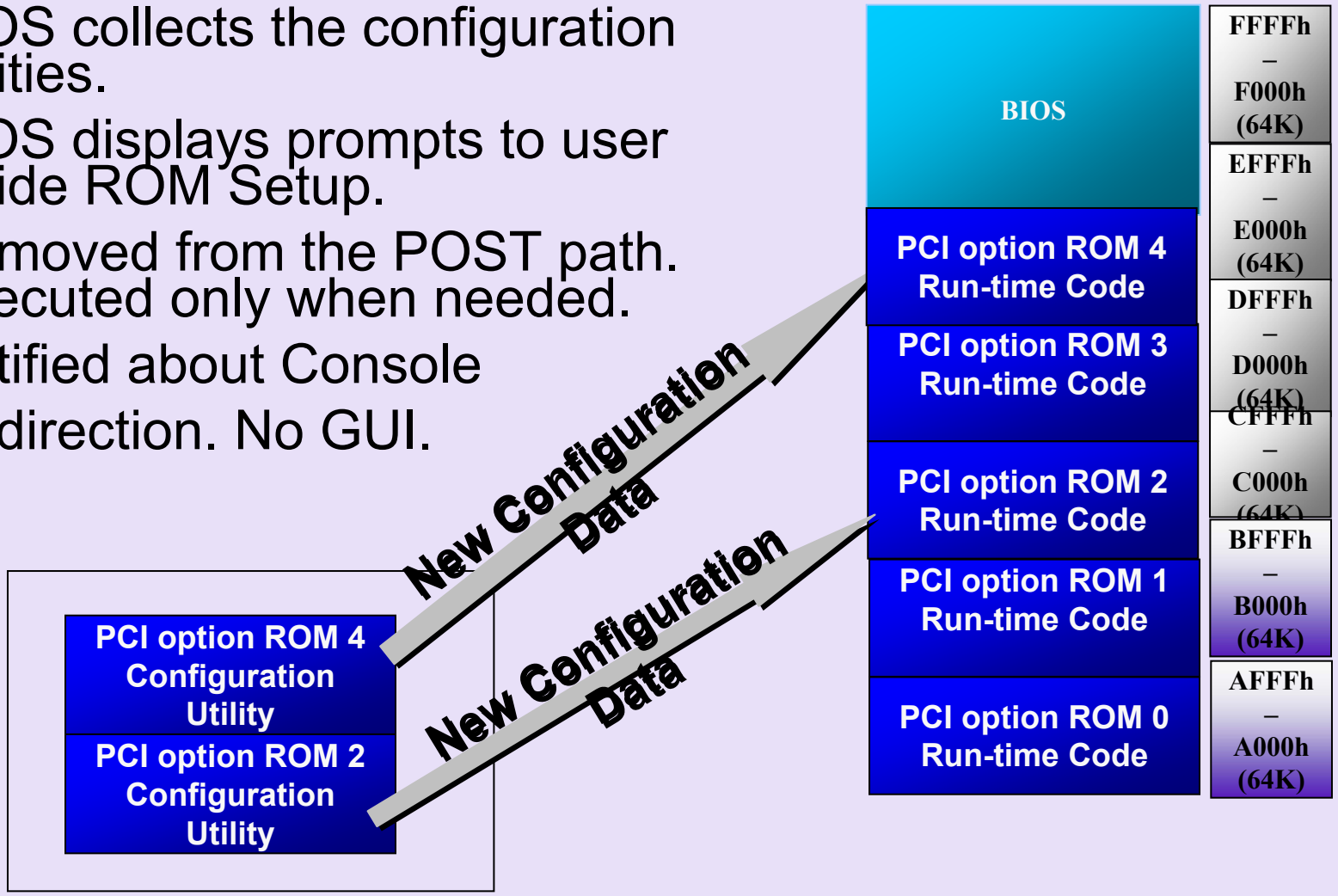
PCI Option ROM Management in the PCI 3.0 Firmware specification

- PCI Option ROMs sometimes include an end-user configuration utility.
- Prompts user to "Press Ctrl-A to enter Configuration Utility". Then waits for a user response.
- PCI 3.0 separates the configuration utility. BIOS saves utility aside for later execution.



PCI Option ROM Management in the PCI 3.0 Firmware specification

- BIOS collects the configuration utilities.
- BIOS displays prompts to user inside ROM Setup.
- Removed from the POST path. Executed only when needed.
- Notified about Console Redirection. No GUI.



PCI Option ROM Management in the PCI 3.0 Firmware specification

- Defines PCI 2.1 behavior more clearly.
- Defines PCI 3.0 Firmware behavior to address the major historical limitations:
  - ✓ More efficient placement of PCI Option ROMs.
  - ✓ Better memory management services from BIOS.
  - ✓ Improved end-user configuration experience.
- Represents an **evolutionary** approach for both BIOS and PCI Option ROM vendors.

PCI Option ROM Management

Trevor Western

Phoenix Technologies Ltd