PCI® Firmware Specification Update

Dong Wei

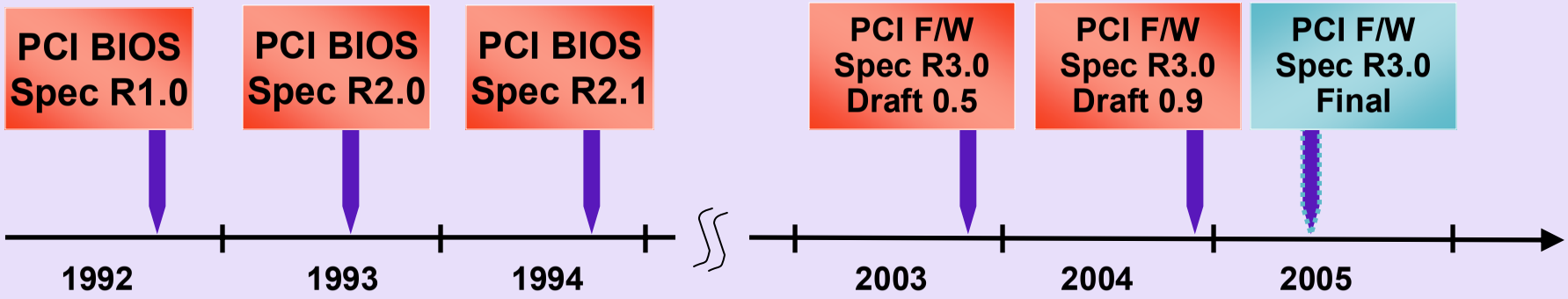
Distinguished Technologist

Hewlett-Packard Co.

Session Outline

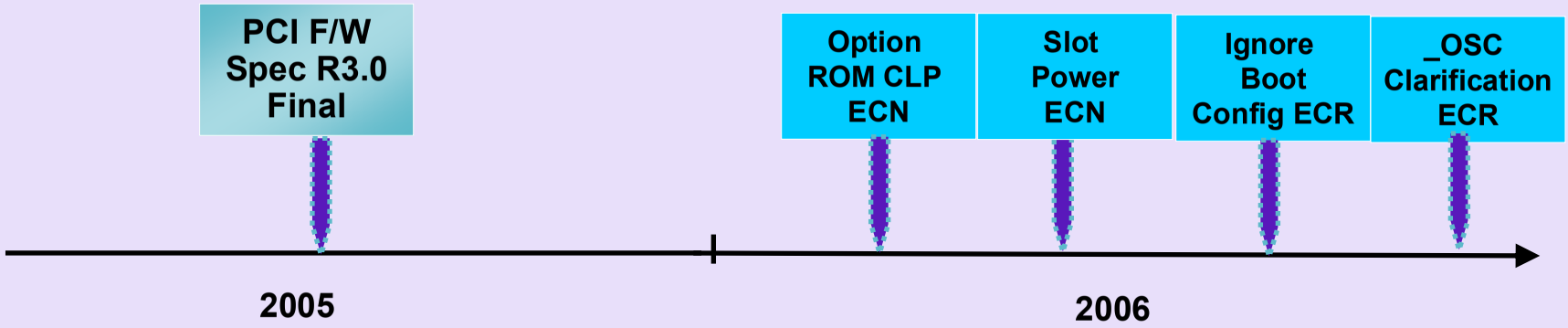
- PCI Firmware Specification Overview
- Option ROM DMTF SM CLP ECN
- Slot Power State ECN
- “Ignore Boot Config” ECR
- PCI _OSC Clarifications ECR

PCI Firmware Spec Evolution



- Started as PCI BIOS Spec back in 1992
- Aligned with PCI Bus Spec 2.0 in 1993 and updated to 2.1 in 1994
- PCI Firmware Specification R3.0 extends PCI BIOS 2.1 spec
  - ✓ Comprehends PCI, PCI-X®, PCI Express® devices and systems
  - ✓ Comprehends Expansion ROM, EFI, ACPI and DIG64 firmware services

PCI Firmware Spec Evolution



- ECRs and ECNs Since PCI Firmware Specification R3.0
  - ✓ Option ROM CLP ECN
  - ✓ Slot Power ECN
  - ✓ Ignore Boot Configuration ECR
  - ✓ _OSC Clarification ECR

PCI Option ROM CLP ECN

- Makes the Command Line Protocol (CLP) calling interface in PCI Firmware Spec R3.0 match DMTF Server Management Command Line Protocol Specification V1.0
  - ✓ API Register Usage Change: EBX, AX, ED:EDI, DS:ESI
  - ✓ Input Arguments
    - AX: Bus/Device/Function, same as ROM INIT and Configuration entry point, indicator as CLP's intended target
    - If entry point supports multiple targets, need to examine the Command Line String Buffer
    - ES:EDI: Pointer to SM CLP Command Line String Buffer
    - DS:ESI: Pointer to SM CLP Command Response String Buffer
  - ✓ Output Arguments
    - AL=0: Success; AL=2: AH has processing error cause; AL=3: EAX[30:16] has execution error code. EAX[31] OEM flag.

PCI Option ROM CLP ECN (Cont.)

- Target IHVs and OEMs
- Defined for PC-Compatible Systems
- Support is optional
  - ✓ Check PCI BIOS Present call
- Equivalent Interfaces for EFI-based systems are defined in the UEFI Specification v2.1 (to be published)
  - ✓ Refer to www.uefi.org

PCI Option ROM CLP ECN (Cont.)

- Power-up Sequence
  - ✓ An SM CLP config session started via a remote console
  - ✓ System FW facilitates the device config by passing SM CLP commands via the SM CLP entry point
  - ✓ Option ROM SM CLP code stores the config setting in a NV location associated with the targeted device to be available to the ROM Init code
  - ✓ System FW initialize the PCI device by calling the ROM Init entry point

PCI Slot Power ECN

- Clarifies the unoccupied slot power state at FW/OS handoff
  - ✓ Allows the unoccupied slots' power to be off at the FW/OS handoff
    - All slots with an open MRL must be disabled and their Power Indicators must be turned off.
    - All occupied slots with MRL closed must be enabled and their Power Indicators must be turned on.
    - The slot power state for the unoccupied slots with MRL closed is IMPLEMENTATION DEPENDENT and their Power indicators must reflect the slot power state.
- No impact to existing HW and SW

Ignore Boot Config ECR

- Added Optional Function 5 to the PCI _DSM Method
  - ✓ System Firmware can indicate to OS that it can ignore the configuration that the firmware has done at boot time, and reconfig/rebalance the resources in the PCI hierarchy.
  - ✓ Return: Integer
    - 1: Ignore Boot Configure
    - 0: OS shall not ignore the PCI configuration that firmware has done at boot time. OS is free to configure the devices in this hierarchy that have not been configured by the firmware.

Ignore Boot Config ECR (Cont.)

- Added Optional Function 5 to the PCI _DSM Method
  - ✓ Placed under
    - virtual PCI-to-PCI bridge object representing the PCIe® root port or switch port,
    - PCI Express endpoint,
    - Conventional PCI/PCI-X bridge,
    - Conventional PCI/PCI-X device.
    - Note: PCI Root Bridge uses _CRS/_PRS/_SRS
  - ✓ When placed on a bridge, the return value applies to the complete hierarchy produced by that bridge.

Ignore Boot Config ECR (Cont.)

- Reasons:
  - ✓ Provides backwards compatibility on platforms that can run legacy operating systems.
  - ✓ The firmware cannot distinguish the OS in time to change the boot configuration of devices.
    - x86 OS in non-PAE mode cannot access device resource space above 4GB. The firmware is required to configure devices at boot time using addresses below 4GB.
    - x64 OS can access device resources above the 4GB so it does not want the firmware to constrain the resource assignment below 4GB.
    - Ignoring the configurations done by firmware at boot time will allow the OS to push resource assignment using addresses above 4GB for an x64 OS while constrain it to addresses below 4GB for an x86 OS.

PCI _OSC Clarification ECR

- Existing:
  - ✓ _OSC applies to the entire hierarchy originated by a PCI Host Bridge
  - ✓ Recommendation: a machine with multiple host bridge devices should report the same capabilities for all host bridges
- Clarifications:
  - ✓ System FW must only mask a Control Field bit to zero if it has explicit knowledge that the feature will not work properly under native operating system control, due to platform errata or other incompatibilities.
  - ✓ Lack of explicit support of a feature in the system FW is not a reason to mask a Control Field bit to zero.

PCI _OSC Clarification ECR (Cont.)

- Clarifications:
  - ✓ "Same Capabilities" Recommendation applies to host bridges of the same type
    - PCI/PCI-X host bridge devices can have one capabilities setting and PCI Express hot bridge devices can have another.
  - ✓ System FW must always assume that PCI/PCI-X features are supported beneath a PCI Express hierarchy
  - ✓ System FW must always assume that PCI Express features are NOT supported beneath a PCI/PCI-X hierarchy

PCI _OSC Clarification ECR (Cont.)

- Clarifications:
  - ✓ FW's "Granting of Control" of Native Hot means that the FW has configured all hot plug events from using the ACPI SCI/GPE route to using PCIe device interrupts, regardless of whether there are hot pluggable slots down the hierarchy.
    - Note: It is OS's responsibility to enable native interrupts prior to calling _OSC
  - ✓ It is the OS's responsibility to determine whether a slot is hot pluggable by examining the status of the slots.

PCI _OSC Clarification ECR (Cont.)

- Implementation Note
  - ✓ Some OSes may require the platform to grant control over all of PCIe features (Hot Plug, PME, AER, Express Capability) in an “ALL or NOTHING” manner.
  - ✓ If control of any one of these capabilities is denied by the platform, such OS may choose not to claim control of any of these features.
  - ✓ Reference: Microsoft WINHEC 2006 Presentations

Call to Action

- Understand and properly use the standard PCI firmware interfaces
  - ✓Review the Backup Slides for the PCI Firmware Spec R3.0 Content
  - ✓Review this Presentation for the new development (ECNs and ECRs) since R3.0
- Effectively use PCI _OSC in the transition from legacy to PCIe-aware OS

Additional Resources

- PCI Firmware Specification R3.0 – http://www.pcisig.com
- Unified EFI Forum – http://www.uefi.org
- ACPI – http://www.acpi.info
- DMTF – http://www.dmtf.org
- DIG64 – http://www.dig64.org
  - ✓ SAL – http://developer.intel.com/design/Itanium/Downloads/245359.htm
- PMM - http://www.phoenix.com/resources/specs-pmm101.pdf
- BIOS Boot Specification – http://www.phoenix.com/resources/specs-bbs101.pdf

- Direct questions on this presentation to : Dong.Wei@hp.com

Backup

Outlines

- Standard PCI Platform Interfaces Common on PC-compatible and DIG64-compliant Systems
  - ✓ EFI Services
  - ✓ ACPI Services
- Standard PCI Platform Interfaces Specific to PC-compatible or DIG64-compliant Systems
  - ✓ MMCFG Space Access
  - ✓ PCI BIOS
  - ✓ Expansion ROM
    - – Extensions for PC-compatible Systems
- Effective Use of _OSC in the Transition from Legacy to PCIe-aware OS

PCI Services in EFI

- PCI Root Bridge Driver
  - ✓ PCI Root Bridge Protocol
    - – Memory, IO, configuration space, bus-mastering DMA
    - – Enable Non-identity MMIO address mapping between host processor view and the PCI device view
- PCI bus driver
  - ✓ PCI IO Protocol
    - – Used by PCI device driver to access memory and IO on a PCI controller
    - – Supports 4GB of configuration space
- PCI device driver
  - ✓ Manages PCI controllers
- EFI uses a single timer interrupt
  - ✓ INTx, MSI, MSI-X not used

Device State at EFI/OS Handoff

- Handoff point
  - ✓ Return from EFI ExitBootServices()
  - ✓ System Firmware (SFW) only required to configure the boot and console devices
    - Configure the entire path (chipset, bridge, multifunction device)
    - Configure all host bridges
    - Not changed by EFI drivers or apps
  - ✓ BAR configuration check
    - Command register: I/O Enable, Memory Enable
    - Enable bit in expansion ROM BAR
      - May share decoders with other BARs, OS cannot assume dual decoders
    - Not the bus master enable bit (may not be able to master a transaction)
  - ✓ SFW to deassert RST# on all occupied slots below HB. This allows for the OS to wait for Trhfa only once

PCI Services in ACPI

- Host bridges are reported in the name space
  - ✓ _HID, _CID (PNP0A03, PNP0A08)
  - ✓ _CRS, may include _TRA, _TTP and _TRS
  - ✓ _PRT
  - ✓ _BBN (required for systems with multiple host bridges)
  - ✓ _UID (required for systems with multiple host bridges)
  - ✓ _SEG, PCI segment group (required for systems supporting multiple PCI segment groups)
    - Supports > 256 bus numbers
  - ✓ _STA
  - ✓ _MAT
  - ✓ _OSC (more on this later)

_DSM

- _DSM enables devices to provide device specific control
- Uniqueness arranged
  - ✓ {UUID, RevID, function index}
- Function 0 is a special query function
  - ✓ Returns which function indices are supported
- Returns from non-zero functions are function-specific

PCI _DSM

- Function 0 is the query function
  - ✓ Return is a buffer
    - Bit 0 is set, bit 1-4 may be set (if the corresponding function is supported).
- Function 1: PCIe slot info
  - ✓ Equivalent to SMBIOS type 9 entry, but with dynamic characteristics
  - ✓ Placed under the virtual PCI-PCI bridge representing RP or switch port that generates the slot on the motherboard
  - ✓ Returns
    - Link width capabilities (x1, x2, …, x16)
    - Type of slot (form factors)
    - Slot signals (SMBus, WAKE)

PCI _DSM (Con't)

- Function 2: PCIe slot number
  - ✓ Source of slot number description
    - Chassis number register & physical slot number
      - Chassis number register: bit 24:31 of the slot numbering capabilities register in PCI-PCI bridge spec)
      - Physical slot number: bit 19:31 of the slot capabilities register (offset 14h in PCI express capability structure)
    - Legacy _SUN
  - ✓ Breaks into tokens
    - Represent hardware
    - Provide user-friendly display
  - ✓ Returns a list of 4-element packages:
    - Source ID
      - _SUN, Chassis ID Reg, Slot Number Field
    - Token ID
      - Chassis, cabinet, IO tray, module, slot, vendor-specific
    - Start bit, End bit

PCI _DSM (Con't)

- Function 2: PCIe Slot Number (Example)

```
Method (_DSM, 3) {
    // Assume GUID and revision match
    Return (Package(3){          // PCIe Slot Parsing
       Package(4){
          1,1,2,7              // bit 7:2 in the PPB Chassis ID
       },                    // Token represents a "Cabinet"
       Package(4){
          1,2,0,1              // bit 1:0 in the PPB Chassis ID
       },                    // Token represents a "I/O Tray"
       Package(4){
          2,4,0,7              // bit 7:0 in the PCIe Physical
                               //Slot Number
       }
    })
}
```

- ✓ If PPB Chassis ID=01001110b, and PCIe Physical Slot Number=17Ah, the Display is "Cabinet 0x13, I/O Tray 2, Slot 0x7A"

PCI _DSM (Con't)

- Function 3: vendor-specific token ID
  - ✓ Arg3 passes in the vendor-specific token ID
  - ✓ Return is a list of 2-element packages
    - Language ID of the string
    - Unicode string representing the token ID
- Function 4: PCI Bus Capability
  - ✓ Report PCI or PCI-X root bus's capabilities and operating mode in a bus capabilities structure
    - Attributes (64-bit, PCI-X mode 1 ECC, device ID messaging)
    - Current speed/mode
    - Supported speeds/modes
    - Voltage
  - ✓ Placed under the PCI or PCI-X host bridge

PCI Slot Description

- Purposes
  - ✓ ACPI-based PCI hot plug
  - ✓ Manageability
- Must list 8 device objects per slot
  - ✓ Each corresponding to the possible 8 functions on the device
- _ADR is used to identify the PCI function address (dev/func number)

Hot Plug Parameters

- _HPP
  - ✓ Evaluates to:
    - Cache-line size, latency timer, SERR enable and PERR enable values
    - Used when configuring a PCI device hot added or not configured by SFW
- _HPX
  - ✓ Setting record type 0 is for PCI, supersedes _HPP
  - ✓ Setting record type 1 is for PCI-X, returns
    - Maximum memory read byte count
    - Average maximum outstanding split transactions
    - Total maximum outstanding split transactions

Hot Plug Parameters (Con't)

- _HPX (Cont.)
  - ✓ Setting record type 2 is for PCIe, will be part of ACPI 3.0 Errata
  - ✓ Placed under RP or switch downstream port
  - ✓ Allows a PCIe-aware OS that supports native hot plug (but does not support features like AER) to program the specified registers of the hot plugged PCIe device
    - (Un)Correctable Error Mask/Severity
    - Advanced Error Capabilities/Control
    - Device Control
    - Link Control
    - Secondary Uncorrectable Error Mask/Severity

Device/Slot State in ACPI-based Hot Plug

- OS is required to configure PCI subsystems during hot plug
- Power state for empty slots follows SHPC
  - ✓ On if MRL closed, off if MRL open
- _PS3 method brings the device to its D3 state
  - ✓ Whether D3hot or D3cold is device dependent
- _EJ0 is used to online remove a PCI device
  - ✓ At _EJ0 completion, the PCI device must be Isolated for physical removal
  - ✓ _EJ0 is recommended to remove the power from the slot if the platform supports it

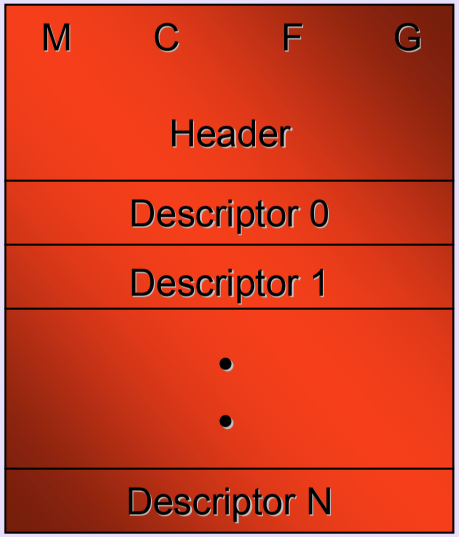
Session Outline

- Standard PCI Platform Interfaces Common on PC-compatible and DIG64-compliant Systems
  - ✓ EFI Services
  - ✓ ACPI Services
- Standard PCI Platform Interfaces Specific to PC-compatible or DIG64-compliant Systems
  - ✓ MMCFG Space Access
  - ✓ PCI BIOS
  - ✓ Expansion ROM
    - – Extensions for PC-compatible Systems
- Effective Use of _OSC in the Transition from Legacy to PCIe-aware OS

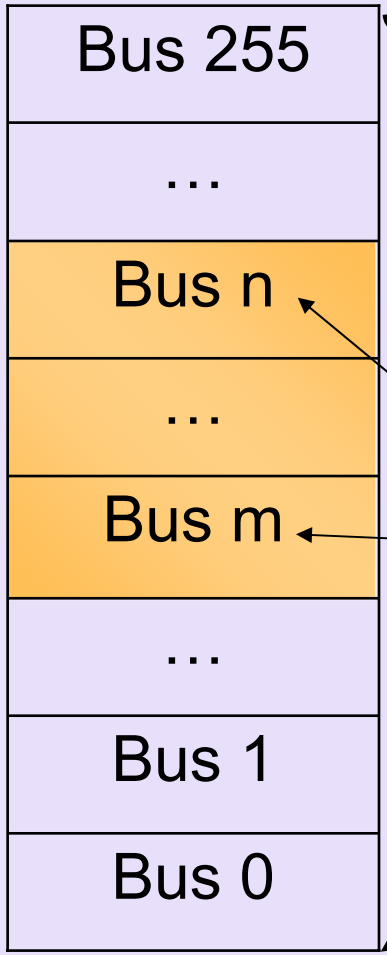
Access to the Enhanced Configuration Space

- On PC-compatible systems:
  - ✓ MCFG (non-hot removable)
  - ✓ _CBA (hot plug capable)
  - ✓ MMCFG Ranges must be reserved by declaring a motherboard resource (PNP0C02)
    - For an OS that does not comprehend MMCFG
  - ✓ Base address is processor-relative
- On DIG64-compliant systems:
  - ✓ SAL_PCI_CONFIG_READ/WRITE procedures shall be used
    - The usage of the segment parameter must match the _SEG
    - These procedures provide completion semantics

MCFG & _CBA



Descriptor



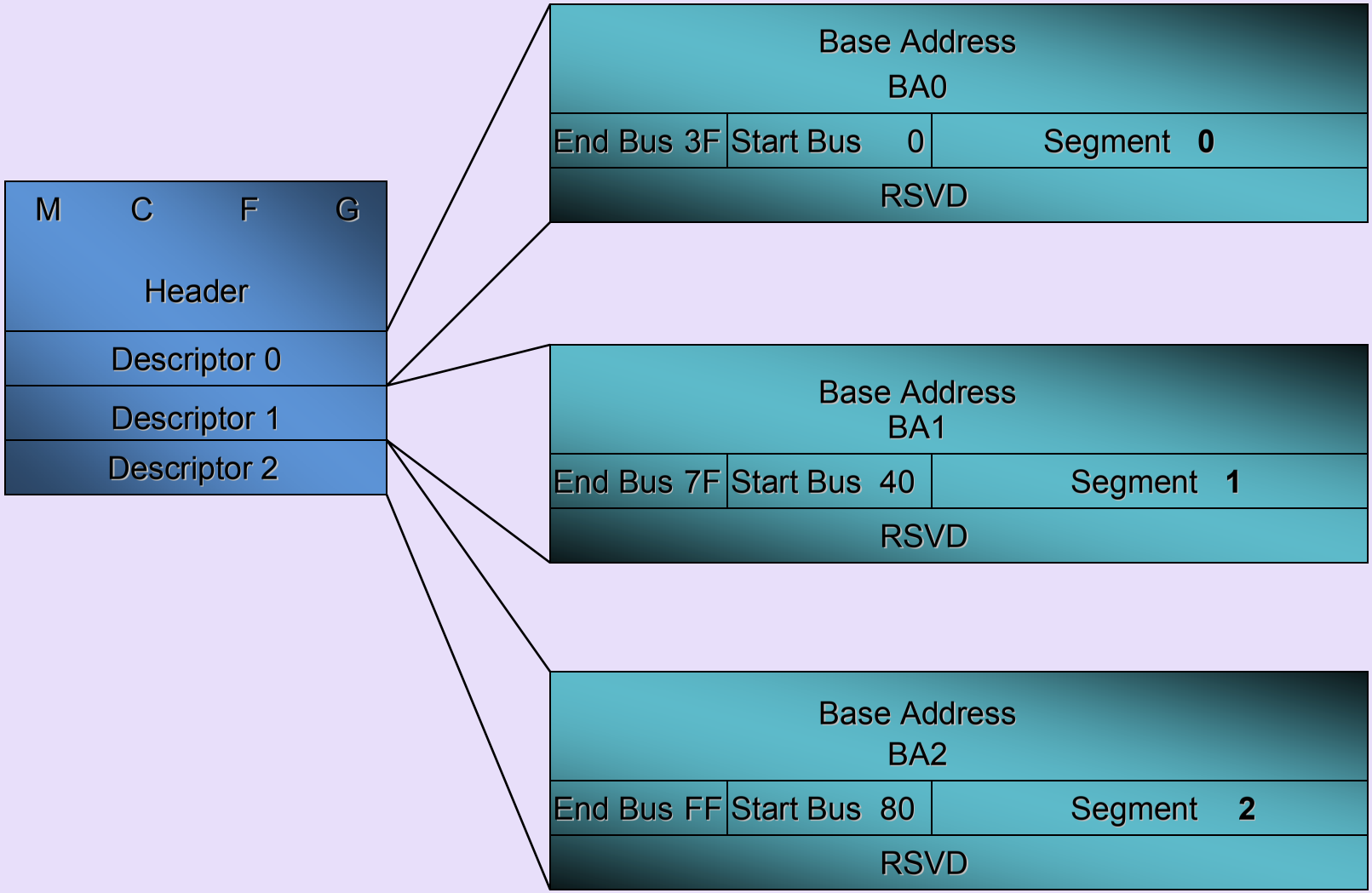
Base Address

Segment Group i

ACPI Namespace

```
Device(PCI1){
  …
  Name(_SEG, …)
  Method(_CRS,
    ResourceTemplate()
  {WORDBusNumber(…)
  }
  Method(_CBA,0){
  …
  }
  …
}
```

Multi-Root = Multi-Segment



PCI BIOS

- Inherited all contents from the PCI BIOS spec
- INT 1Ah calls are used during POST in real mode
  - ✓ After POST, Use MCFG, INT 1Ah may not be supported
- Added Bits in CH for PCI BIOS present function
  - ✓ Enables enhanced configuration space access
  - ✓ Supports standardized configuration code execution
  - ✓ Supports DMTF CLP style configuration

PCI Expansion ROM

- Definition moved from the conventional PCI specification
- New in 3.0 for x86 expansion ROM
  - ✓ Must verify the system BIOS is 3.0 compliant
  - ✓ Adds INIT arguments
    - Run-time address (useful if space is not enough for initialization)
    - Segment/Offset of the PMM services entry point
  - ✓ Must use PMM to reserve memory
  - ✓ Adds maximum run-time image length in the PCI data structure
  - ✓ Expands the x86 expansion ROM region to A000h-FFFFh
  - ✓ Lifts the C000h initialization address requirement for VGA expansion ROM

PCI Expansion ROM (con't)

- New to 3.0 for x86 Expansion ROM (Con't)
  - ✓ Address alignment now 512-byte (vs. 2KB)
  - ✓ Must support BIOS boot specification
  - ✓ Interrupt handler chain to the next, no EOI
  - ✓ Add standard support for configuration code execution
    - Do not change video mode when redirecting console
  - ✓ Add standard support for configuration via DMTF CLP

Session Outline

- Standard PCI Platform Interfaces Common on PC-compatible and DIG64-compliant Systems
  - ✓ EFI Services
  - ✓ ACPI Services
- Standard PCI Platform Interfaces Specific to PC-compatible or DIG64-compliant Systems
  - ✓ MMCFG Space Access
  - ✓ PCI BIOS
  - ✓ Expansion ROM
    - – Extensions for PC-compatible Systems
- Effective Use of _OSC in the Transition from Legacy to PCIe-aware OS

_OSC

- Arguments
  - ✓ Arg0 (Buffer): UUID
  - ✓ Arg1 (Integer): Revision ID
  - ✓ Arg2 (Integer): Count of Capabilities DWORDs
  - ✓ Arg3 (Buffer): Capabilities DWORDs
- Return Code
  - ✓ Capabilities DWORDs in Buffer

PCI _OSC

- Placed under host bridge
  - ✓ Required for PCIe, optional for PCI and PCI-X
- Used by OSPM to
  - ✓ Query the device capabilities
  - ✓ Communicate the device driver capability support
- Capabilities DWORDs
  - ✓ First: generic DWORD
    - – Provides query and status/error
  - ✓ Second: support field DWORD
    - – Informs FW the OS supported capabilities
  - ✓ Third: control field DWORD
    - – OS requests of control, FW can reject
    - – Typically involve interrupts/events redirection

Generic DWORD

- Passed in as First DWORD in Arg3
  - ✓ Bit 0 - Query Support Flag
    - When Set, Query the Control Field DWORD Bits
    - When Query, FW does not Change HW Setting
    - OS Must Query until Capabilities Mask (Bit 4 in Return) is Clear
- When return,
  - ✓ Bit 0 - Reserved (not used)
  - ✓ Bit 1 - _OSC failure.
  - ✓ Bit 2 - Unrecognized UUID.
  - ✓ Bit 3 - Unrecognized Revision.
  - ✓ Bit 4 - Capabilities Masked.
  - ✓ All others - reserved.

Support Field DWORD

- Passed in as Second DWORD in Arg3
- Bit Definitions
  - ✓ Bit 0 – if OS Supports ASL Access thru Enhanced Configuration Space
  - ✓ Bit 1 – if OS Natively Supports Active State Power Management Registers in PCIe
  - ✓ Bit 2 – if OS Supports the Clock Power Management Capability and will Enable it during a Native Hot Plug Insertion Event, if Supported by the Added Device
  - ✓ Bit 3 – if OS Supports PCI Segment Group
  - ✓ Bit 4 – if OS supports Configuration of Devices to Generate MSI or MSI-X

Control Field DWORD

- Passed in as Third DWORD in Arg3
- Bit Definitions
  - ✓ Bit 0 – Request Control over PCIe Native Hot Plug
  - ✓ Bit 1 – Request Control over PCI/PCI-X SHPC Hot Plug, Supersedes OSHP
  - ✓ Bit 2 – Request Control over PCIe Native PMEs
  - ✓ Bit 3 – Request Control over PCIe AER
  - ✓ Bit 4 – Request Control over the PCIe Capability Structures
    - PCIe Capability
    - Virtual Channel Extended Capability
    - Power Budgeting Extended Capability
    - AER Extended Capability
    - Serial Number Extended Capability

PCI _OSC Evaluation Rules

- The OS must evaluate _OSC before evaluating any other device specific objects.
- If the _OSC method is not present, then the OS must **not** enable or attempt to use the feature.
- The OS must evaluate _OSC at
  - ✓ Driver native capability support initialization
  - ✓ Notify(device, 8) received by PCI host bridge device
  - ✓ Upon resume from S4
- The OS is permitted to evaluate _OSC any number of times
  - ✓ OS shall not relinquish control of a feature previously requested and granted. OS must preserve the declared and granted capabilities.
- Firmware may not reject already-granted capability control