EDK II Topology

PCI Enumeration

All core package code referenced in this document is located in the **GitHub EDK II repository**.

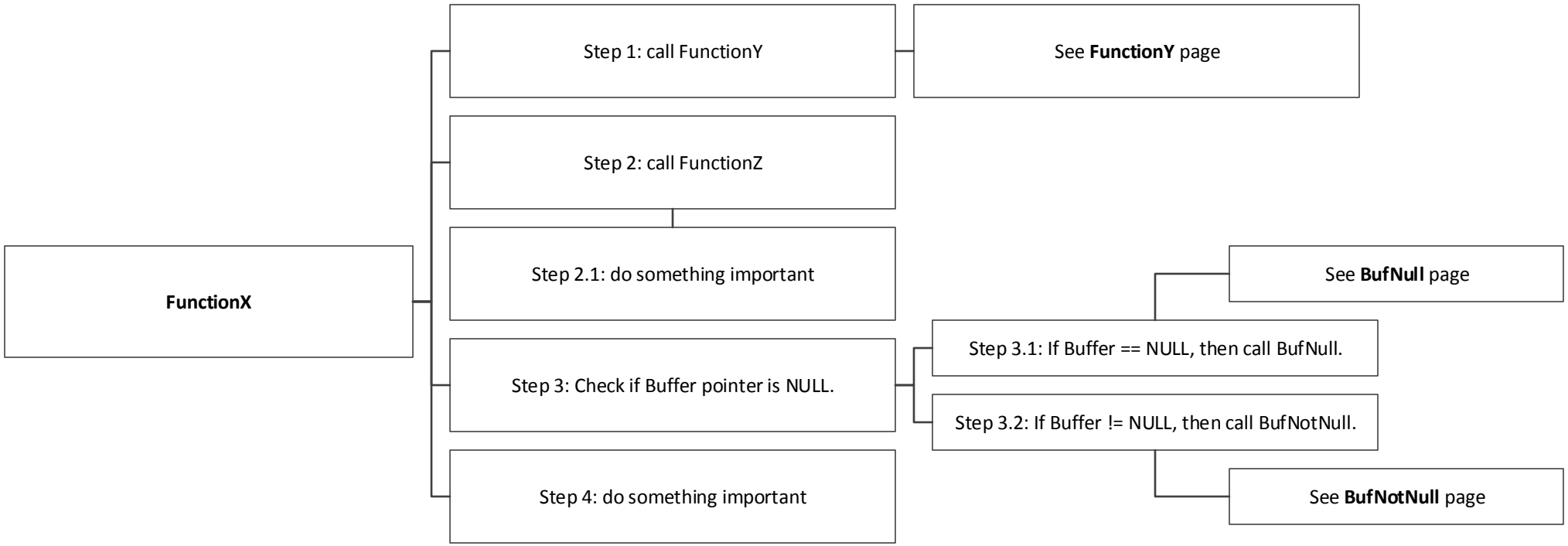
For more in-depth information about EDK II, visit the **Intel® Firmware: Beyond BIOS** page.

Visit TianoCore.org for more **EDK II documentation** and **EDK II projects.**

The function being discussed always starts in the only box on the far left. Boxes represent steps in a function, a branch evaluation in a funtion, or a note to see details on another page about a function being called at a step. Connectors between boxes indicate code flow (who called what) and should be read left-to-right, top-to-bottom. Text in each box will indicate if it's a call, a branch evaluation, or a note. This format was chosen to fit important function details on 1 page.

For the example below, the equivalent C code (note connectors from Step 1-4 to FunctionX):

```
FunctionX() {
        FunctionY();    // see details on FunctionY page
        FunctionZ();    // Step 2; Step 2.1 is in FunctionZ() and is listed because it is important; note the connector between Step 2 and 2.1
        If (Buffer == NULL)     // Step 3
                BufNull();      // Step 3.1
        else
                BufNotNull();   // Step 3.2
        Step 4
}
```



How protocol services are defined so one can find the protocol code to examine it

Summary: if you want to examine the code for a protocol function, you should find the structure definition for the protocol, then find the declaration of the structure, then find the structure member that corresponds with the protocol in the structure definition because they may have different names. EFI_BOOT_SERVICES defines LocateProtocol, mBootServices is of type EFI_BOOT_SERVICES, and the structure member CoreLocateProtocol corresponds with the structure definition LocateProtocol.

MdePkg\Include\UefiSpec.h defines EFI_BOOT_SERVICES structure, and has structure members for protocol services (LocateProtocol, InstallProtocolInterface, etc). MdeModulePkg\Core\Dxe\DxeMain.c has a variable mBootServices of type EFI_BOOT_SERVICES. mBootServices sets function pointers for functions such as LocateProtocol to CoreLocateProtocol and InstallMultipleProtocolInterfaces to CoreInstallMultipleProtocolInterfaces. These functions are defined in MdeModulePkg\Core\Dxe\Handle.c.

How protocols are loaded from flash into memory

Summary: drivers are loaded from flash into memory by some mechanisms into a linked list during the PEI phase.

MdeModulePkg\Core\Dxe\DxeMain.c has DxeMain function which is called when the DXE Core driver is loaded. MdeModulePkg\Core\Dxe\DxeMain.inf has MODULE_TYPE=DXE_CORE and ENTRY_POINT=DxeMain. The end of DxeMain calls CoreInstallMultipleProtocolInterface with the GUID for the HOB that was populated with drivers from the flash part during PEI. PEI phase calls ReadSection (associated with FvReadFileSection in Universal\FirmwareVolume\FwVolDxe\FwVol.c), which eventually gets to a call to LocateProtocol with gEfiDecompressProtocolGuid as a parameter.

Platform DSC libraries for PCI

This is a partial list of libraries that platform DSC file typically uses for PCI.

- IoLib|MdePkg/Library/BaseIoLibIntrinsic/BaseIoLibIntrinsic.inf
- PciLib|MdePkg/Library/BasePciLibCf8/BasePciLibCf8.inf
- PciCf8Lib|MdePkg/Library/BasePciCf8Lib/BasePciCf8Lib.inf
- PciExpressLib|MdePkg/Library/BasePciExpressLib/BasePciExpressLib.inf
- S3PciLib|MdePkg/Library/BaseS3PciLib/BaseS3PciLib.inf

Platform DSC drivers for PCI host bridge initialization

This is a partial list of drivers that platform DSC file must use to initialize PCI services.

- MdeModulePkg/Universal/PcatSingleSegmentPciCfg2Pei/PcatSingleSegmentPciCfg2Pei.inf
- MdeModulePkg/Bus/Pci/PciBusDxe/PciBusDxe.inf (often overridden)
- PcAtChipsetPkg/PciHostBridgeDxe/PciHostBridgeDxe.inf (often overridden)
- DXE_DRIVER that sets EFI_PCI_PLATFORM_PROTOCOL items (PCI Callback, PCI Phase Notify, etc).

See **PI Volume 1 "Additional PPIs" PCI Configuration PPI** for details.

Platform DSC PCDs for PCI

This is a partial list of PCDs that platform DSC file typically sets for PCI.

- gEfiMdePkgTokenSpaceGuid.PcdPciExpressBaseAddress|$(MMCFG_BASE_ADDRESS)
- gEfiMdePkgTokenSpaceGuid.PcdPciDisableBusEnumeration

PCI Driver Binding protocols that are installed

Start:

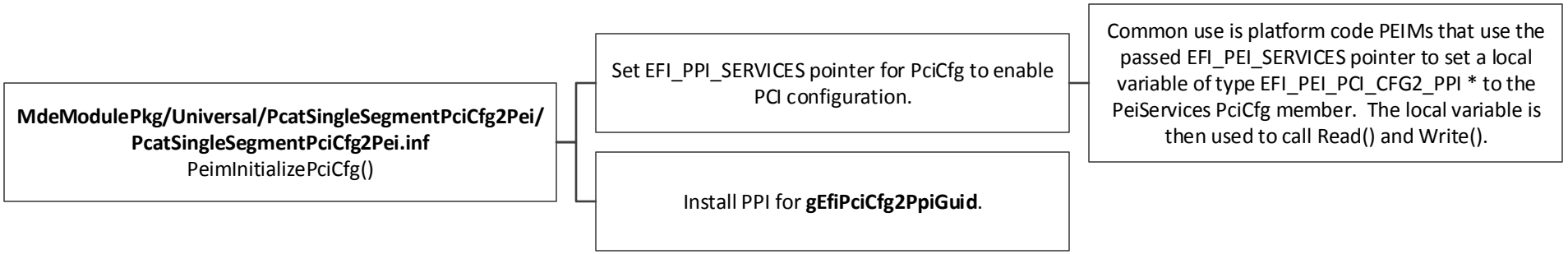
- **gEfiIncompatiblePciDeviceSupportProtocolGuid**
- **gEfiPciPlatformProtocolGuid**
- **gEfiPciOverrideProtocolGuid**

See **PI Volume 5 "PCI Platform"** for details.

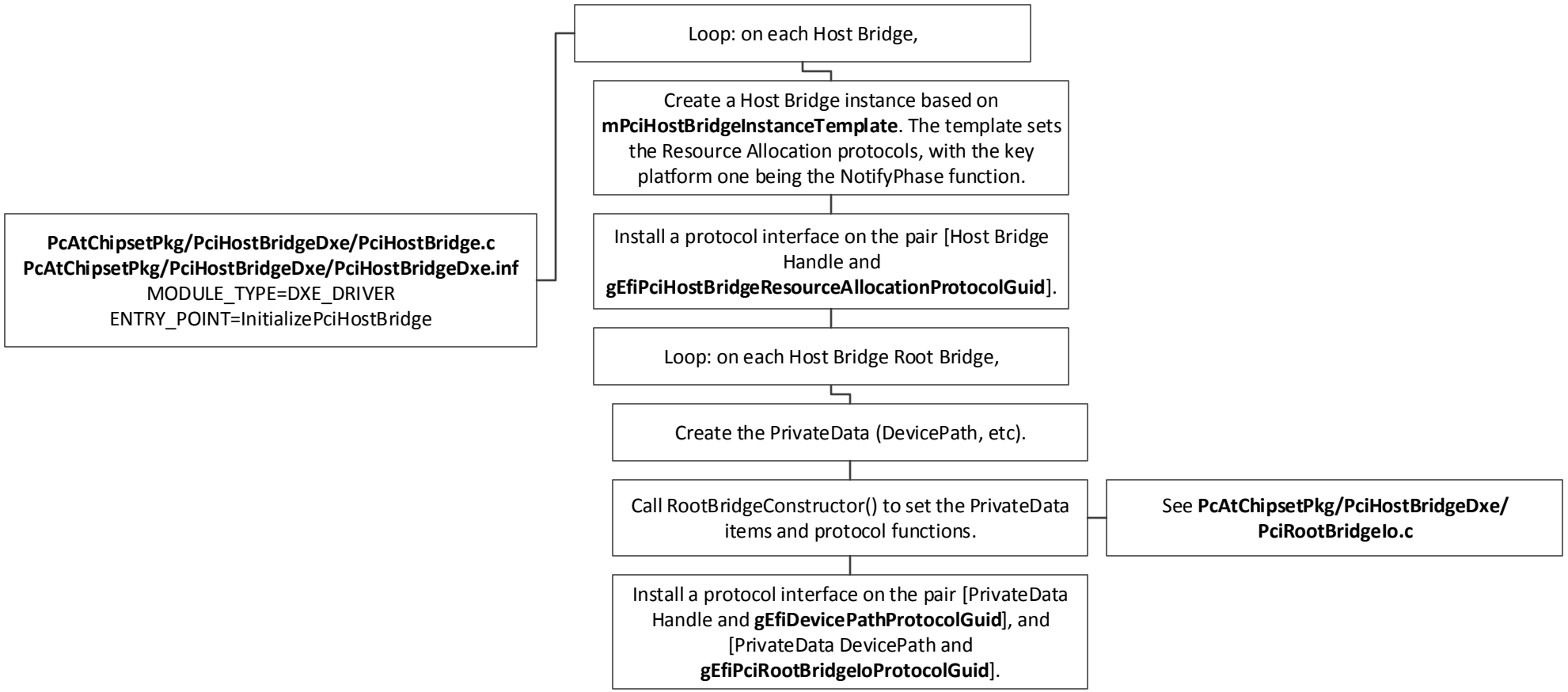
PCI Enumeration Notification Phases

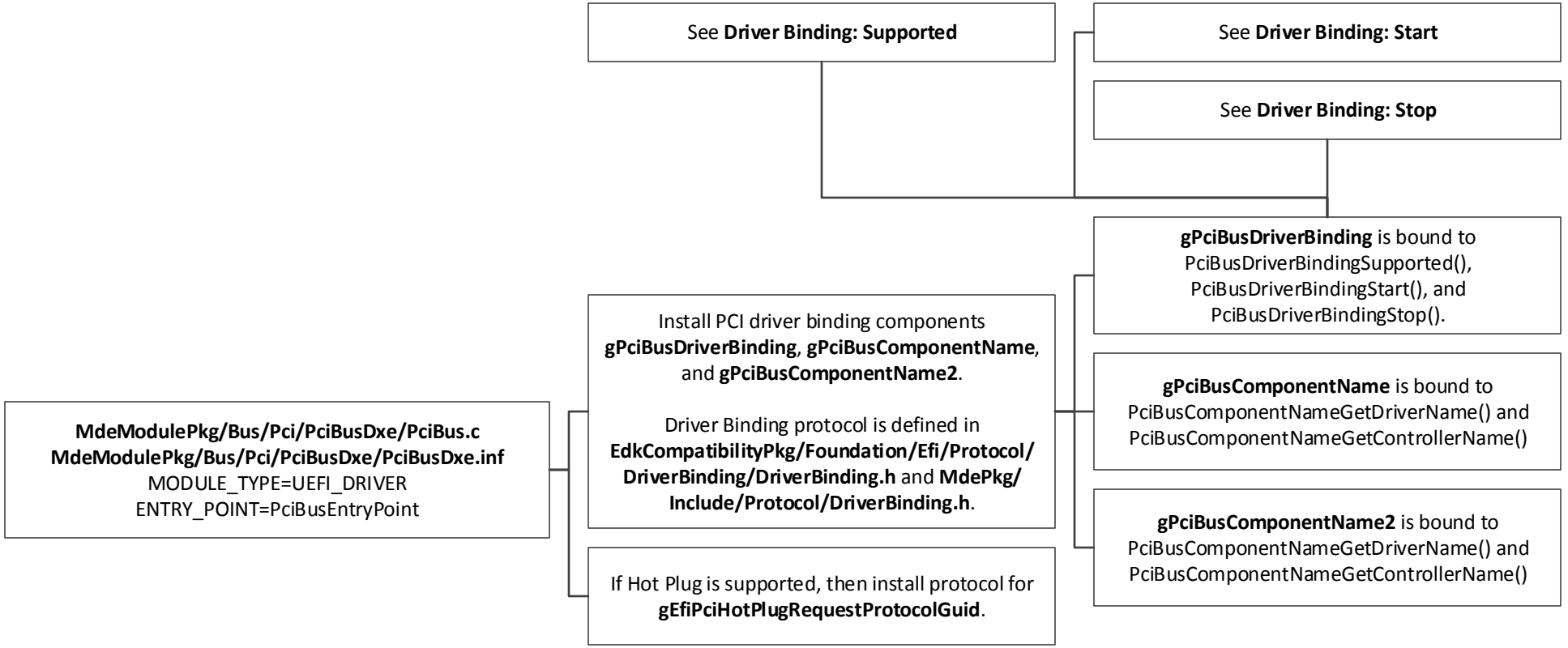
Platforms typically override the PciHostBridgeDxe driver to customize PCI enumeration phases.

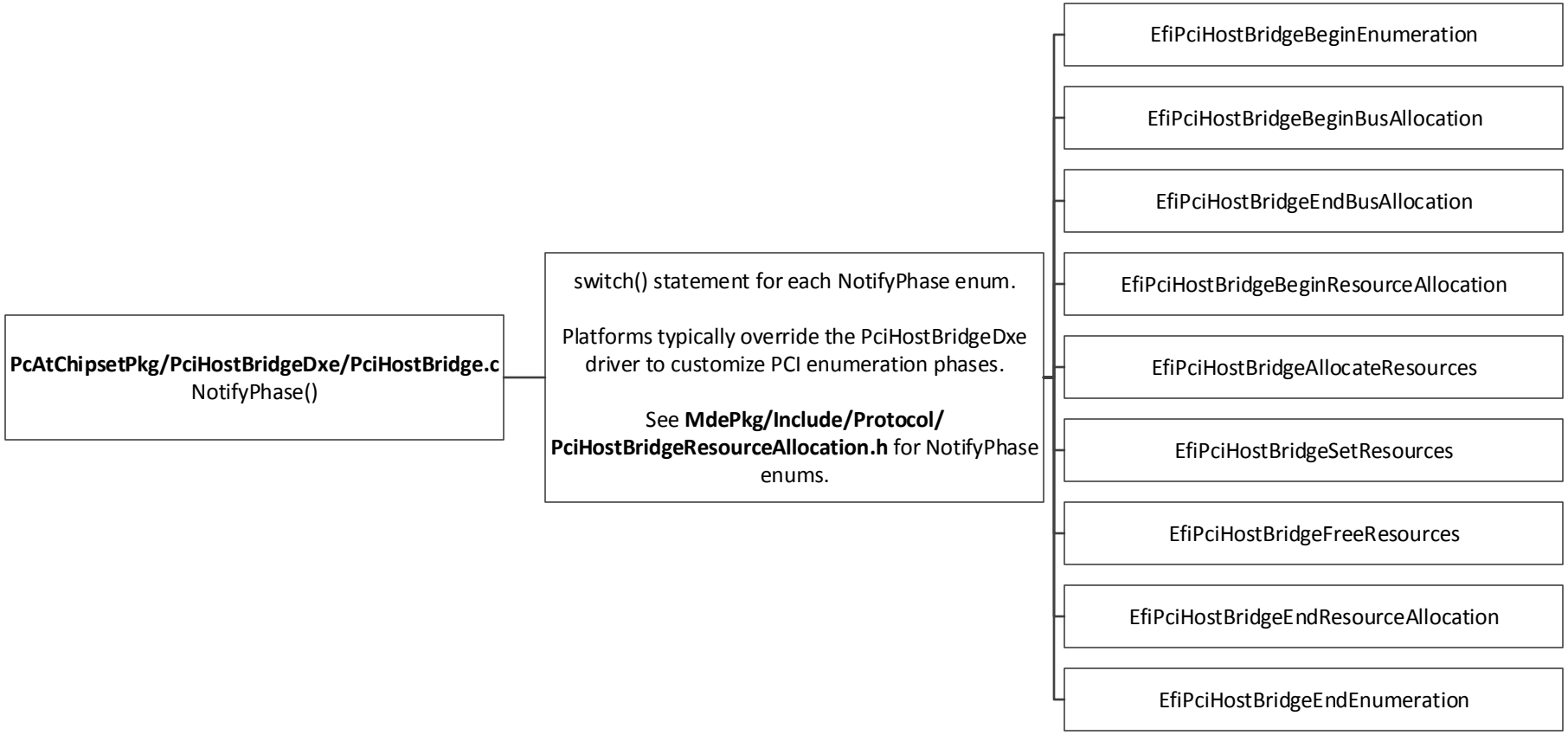
See **NotifyPhase**

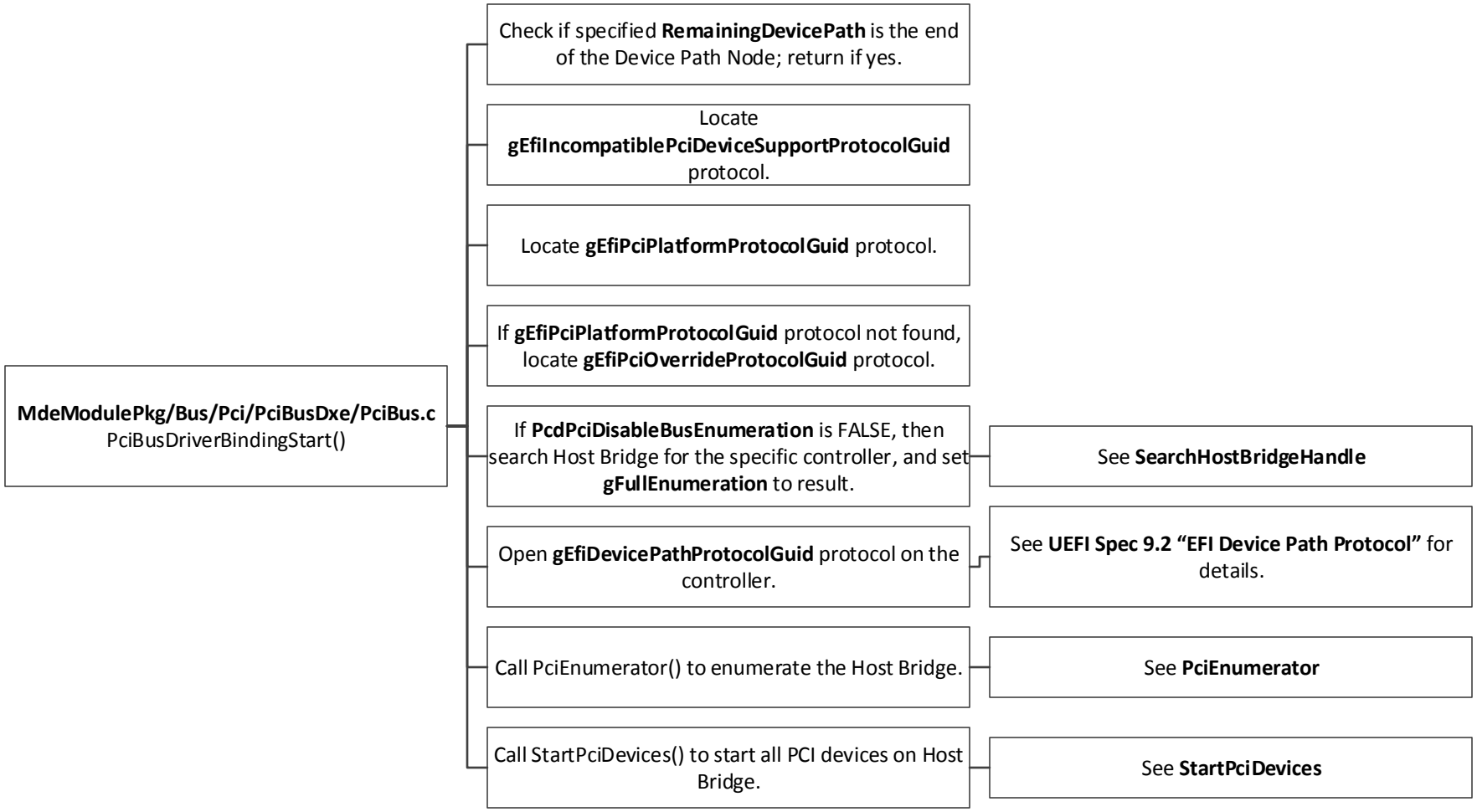

MdeModulePkg/Universal/PcatSingleSegmentPciCfg2Pei/
PcatSingleSegmentPciCfg2Pei.inf
PeimInitializePciCfg()
Set EFI_PPI_SERVICES pointer for PciCfg to enable PCI configuration.
Install PPI for gEfiPciCfg2PpiGuid.
Common use is platform code PEIMs that use the passed EFI_PEI_SERVICES pointer to set a local variable of type EFI_PEI_PCI_CFG2_PPI * to the PeiServices PciCfg member. The local variable is then used to call Read() and Write().

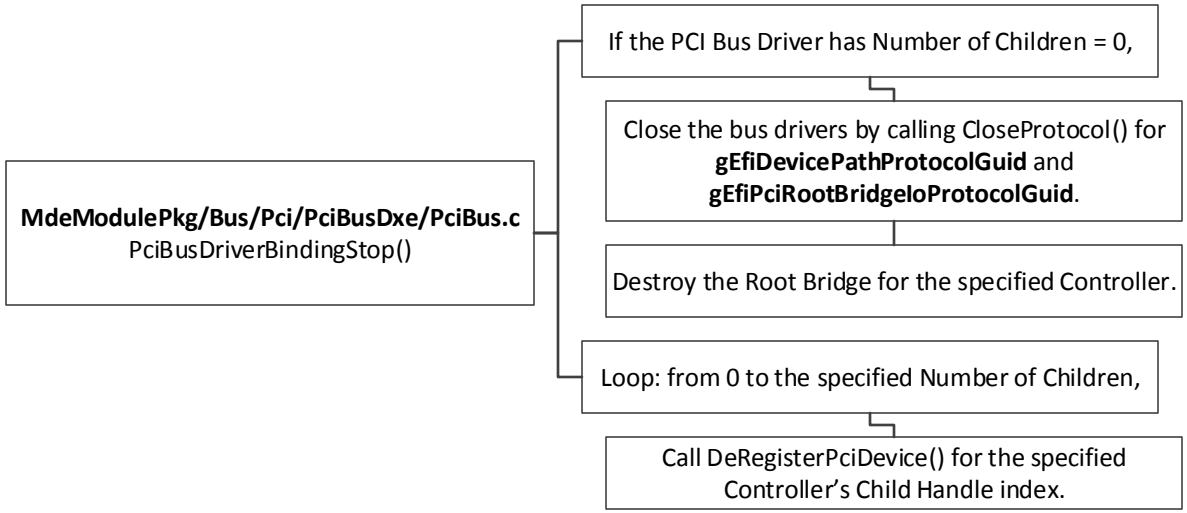
See **Platform** for steps the platform code must take to initialize PCI.

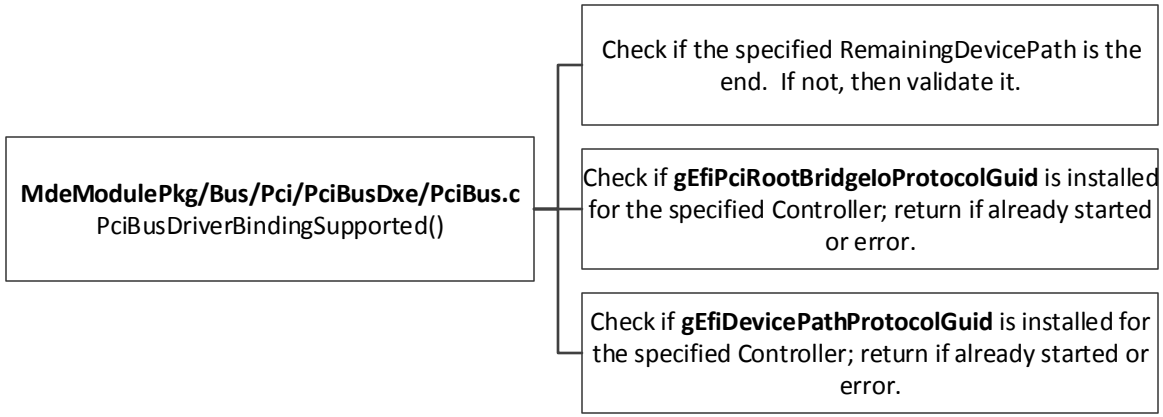


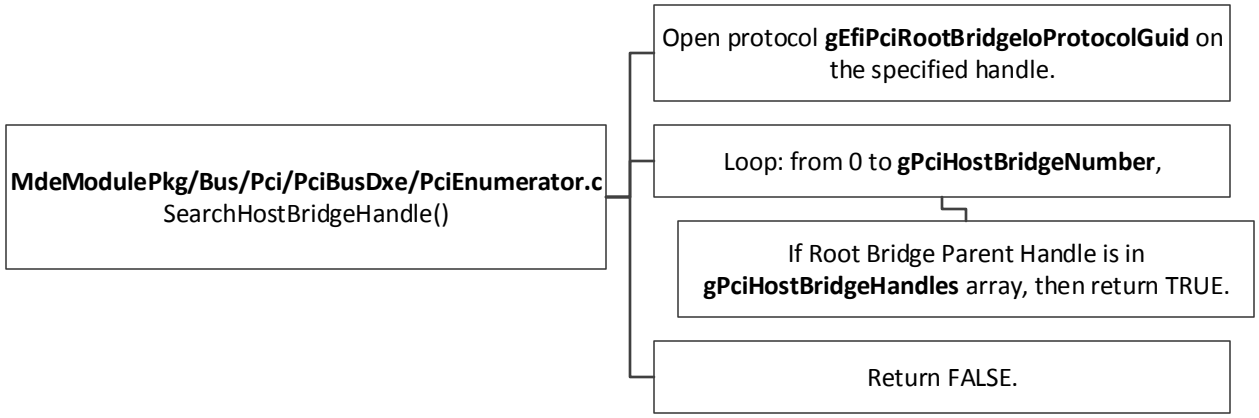

See **Driver Binding: Supported**
See **Driver Binding: Start**
See **Driver Binding: Stop**
gPciBusDriverBinding is bound to PciBusDriverBindingSupported(), PciBusDriverBindingStart(), and PciBusDriverBindingStop().
Install PCI driver binding components **gPciBusDriverBinding**, **gPciBusComponentName**, and **gPciBusComponentName2**.
Driver Binding protocol is defined in **EdkCompatibilityPkg/Foundation/Efi/Protocol/DriverBinding/DriverBinding.h** and **MdePkg/Include/Protocol/DriverBinding.h**.
gPciBusComponentName is bound to PciBusComponentNameGetDriverName() and PciBusComponentNameGetControllerName()
gPciBusComponentName2 is bound to PciBusComponentNameGetDriverName() and PciBusComponentNameGetControllerName()
MdeModulePkg/Bus/Pci/PciBusDxe/PciBus.c
MdeModulePkg/Bus/Pci/PciBusDxe/PciBusDxe.inf
MODULE_TYPE=UEFI_DRIVER
ENTRY_POINT=PciBusEntryPoint
If Hot Plug is supported, then install protocol for **gEfiPciHotPlugRequestProtocolGuid**.

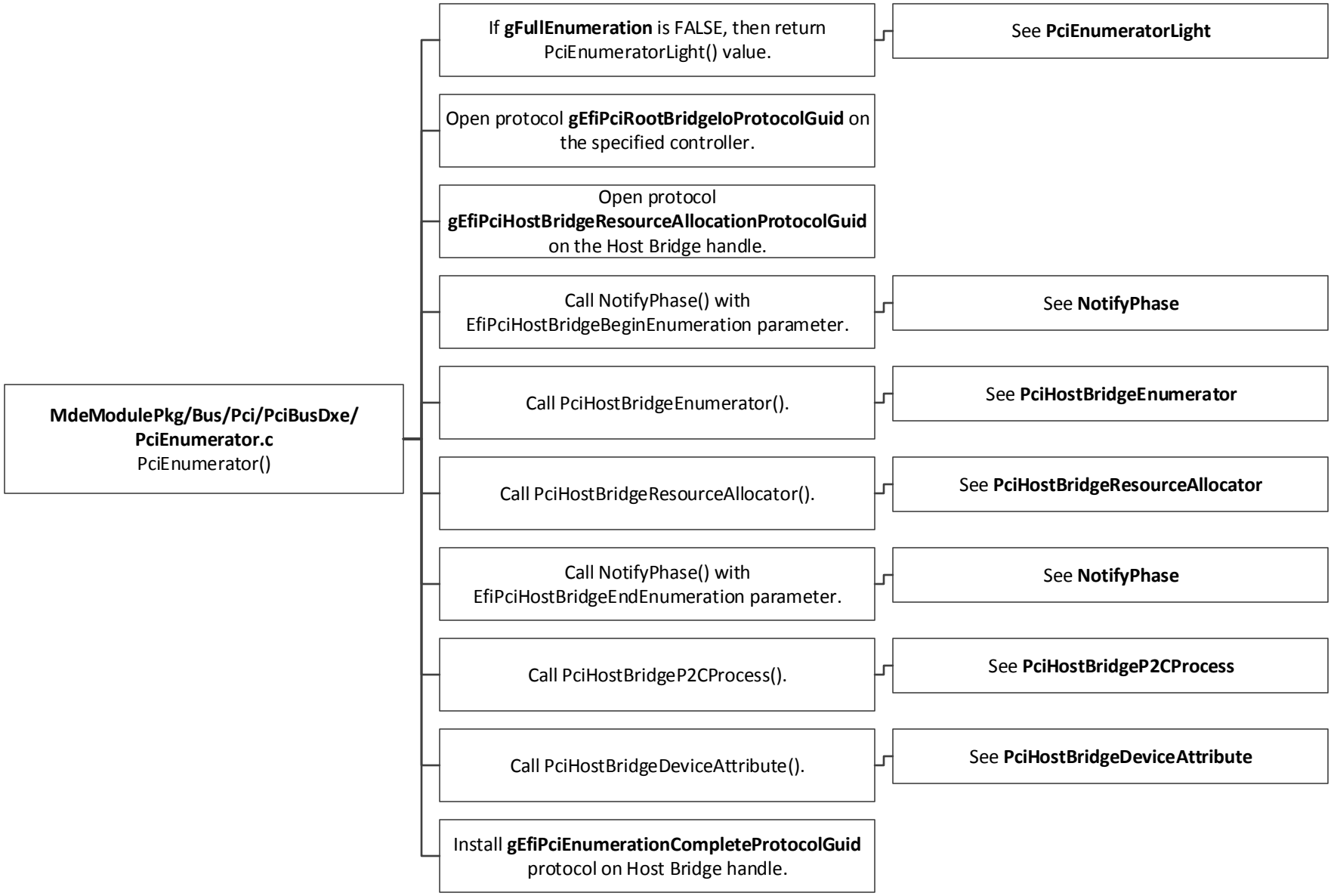

PcAtChipsetPkg/PciHostBridgeDxe/PciHostBridge.c
NotifyPhase()
switch() statement for each NotifyPhase enum.
Platforms typically override the PciHostBridgeDxe driver to customize PCI enumeration phases.
See MdePkg/Include/Protocol/ PciHostBridgeResourceAllocation.h for NotifyPhase enums.
EfiPciHostBridgeBeginEnumeration
EfiPciHostBridgeBeginBusAllocation
EfiPciHostBridgeEndBusAllocation
EfiPciHostBridgeBeginResourceAllocation
EfiPciHostBridgeAllocateResources
EfiPciHostBridgeSetResources
EfiPciHostBridgeFreeResources
EfiPciHostBridgeEndResourceAllocation
EfiPciHostBridgeEndEnumeration

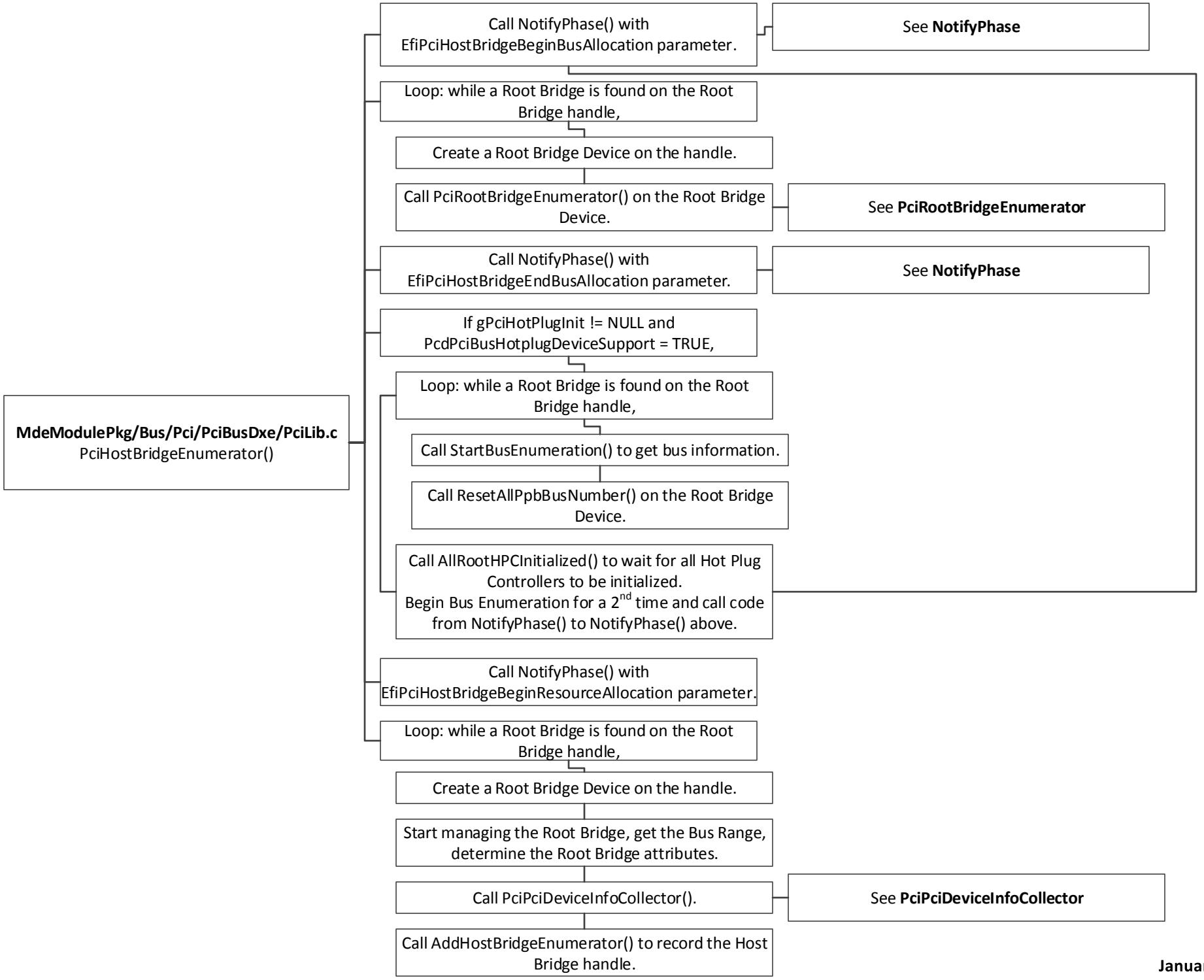

MdeModulePkg/Bus/Pci/PciBusDxe/PciBus.c
PciBusDriverBindingStart()
Check if specified RemainingDevicePath is the end of the Device Path Node; return if yes.
Locate gEfiIncompatiblePciDeviceSupportProtocolGuid protocol.
Locate gEfiPciPlatformProtocolGuid protocol.
If gEfiPciPlatformProtocolGuid protocol not found, locate gEfiPciOverrideProtocolGuid protocol.
If PcdPciDisableBusEnumeration is FALSE, then search Host Bridge for the specific controller, and set gFullEnumeration to result.
See SearchHostBridgeHandle
Open gEfiDevicePathProtocolGuid protocol on the controller.
See UEFI Spec 9.2 "EFI Device Path Protocol" for details.
Call PciEnumerator() to enumerate the Host Bridge.
See PciEnumerator
Call StartPciDevices() to start all PCI devices on Host Bridge.
See StartPciDevices

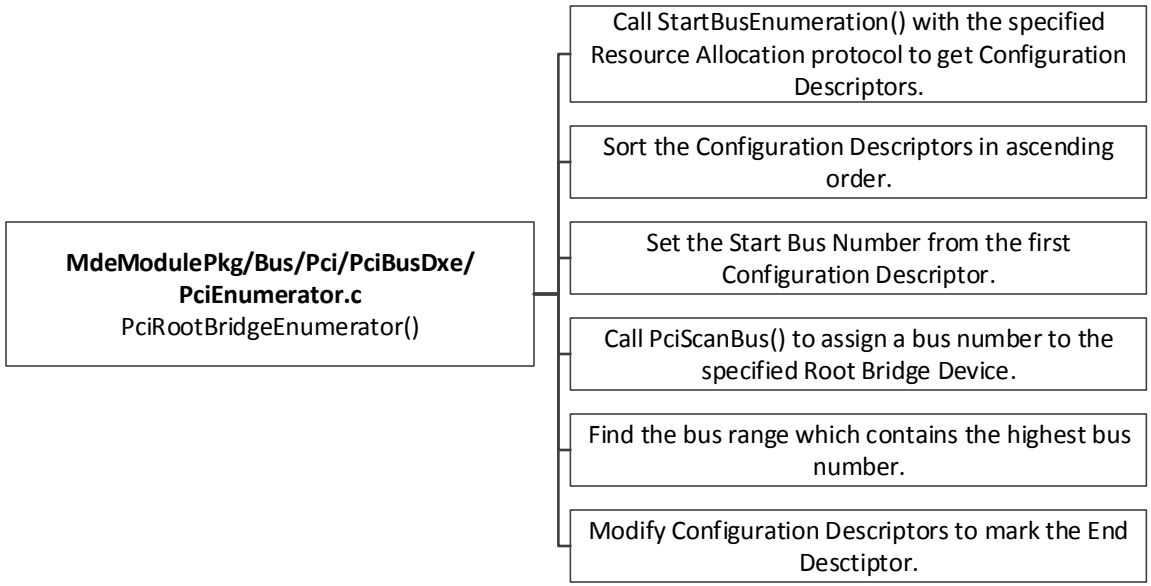

MdeModulePkg/Bus/Pci/PciBusDxe/PciBus.c
PciBusDriverBindingStop()
If the PCI Bus Driver has Number of Children = 0,
Close the bus drivers by calling CloseProtocol() for **gEfiDevicePathProtocolGuid** and **gEfiPciRootBridgeIoProtocolGuid**.
Destroy the Root Bridge for the specified Controller.
Loop: from 0 to the specified Number of Children,
Call DeRegisterPciDevice() for the specified Controller's Child Handle index.

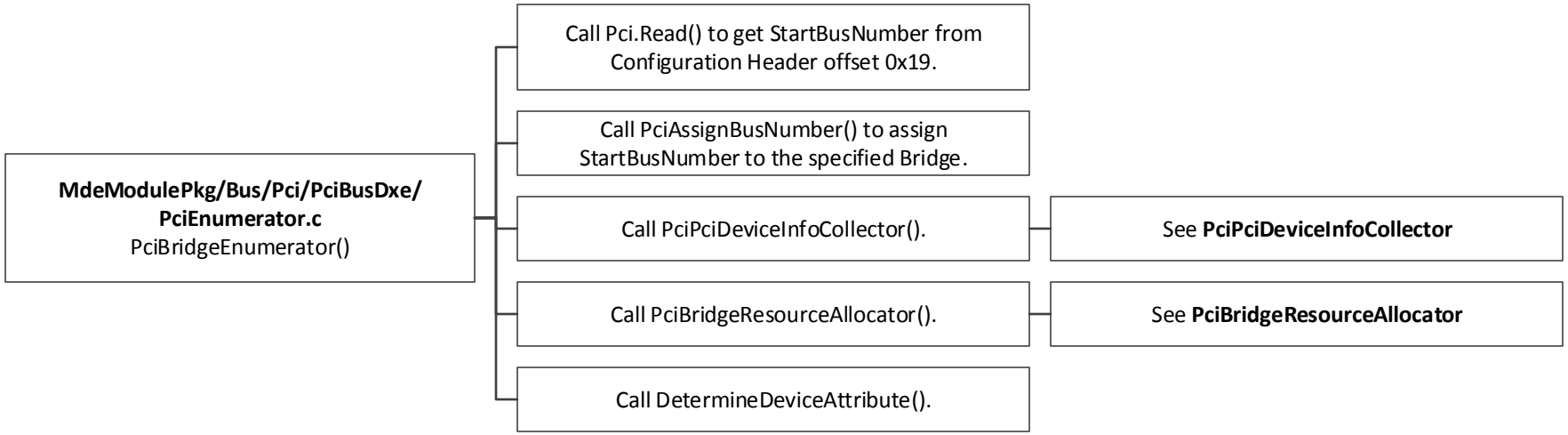

MdeModulePkg/Bus/Pci/PciBusDxe/PciBus.c
PciBusDriverBindingSupported()
Check if the specified RemainingDevicePath is the end. If not, then validate it.
Check if **gEfiPciRootBridgeIoProtocolGuid** is installed for the specified Controller; return if already started or error.
Check if **gEfiDevicePathProtocolGuid** is installed for the specified Controller; return if already started or error.

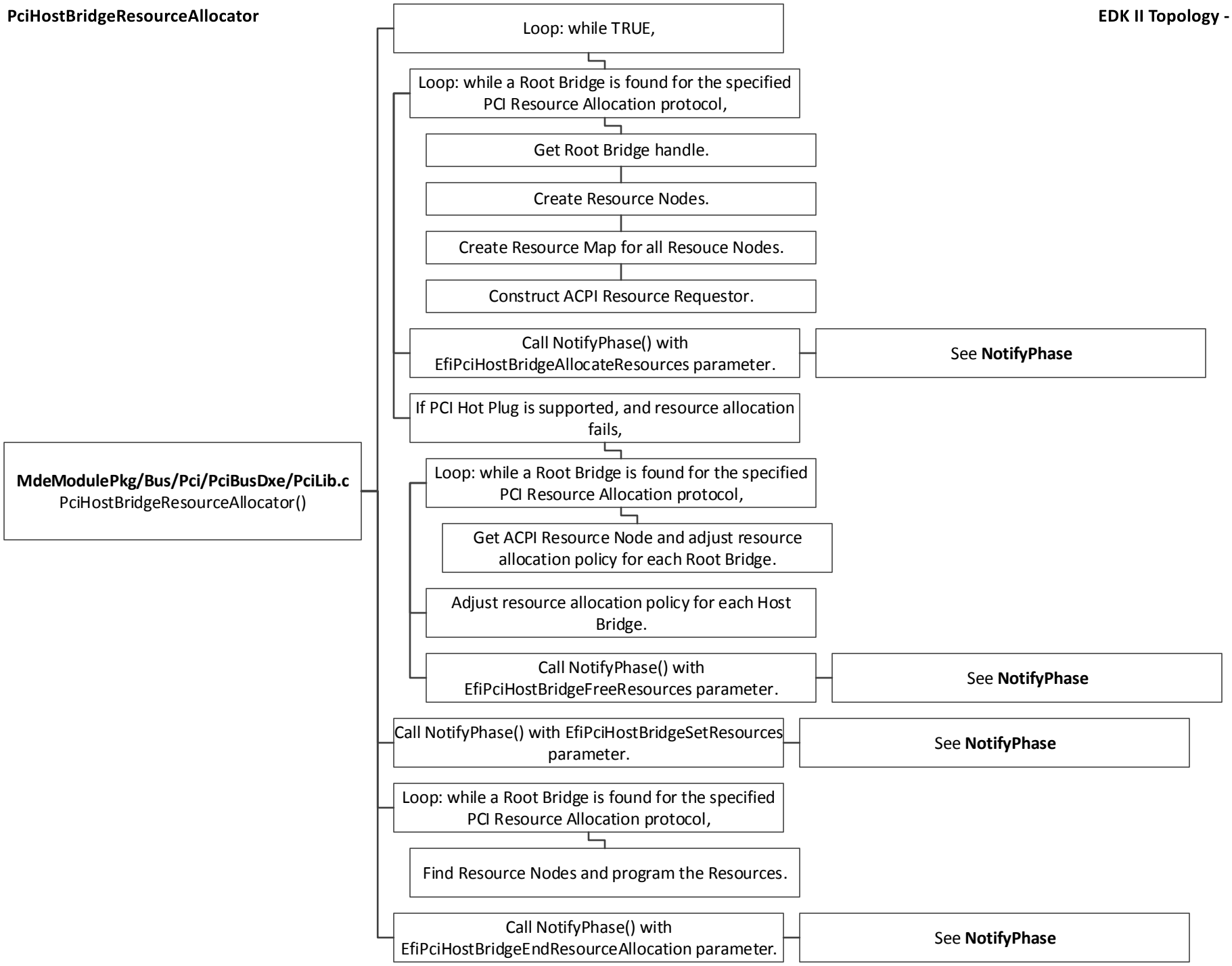

MdeModulePkg/Bus/Pci/PciBusDxe/PciEnumerator.c
SearchHostBridgeHandle()
Open protocol gEfiPciRootBridgeIoProtocolGuid on the specified handle.
Loop: from 0 to gPciHostBridgeNumber,
If Root Bridge Parent Handle is in gPciHostBridgeHandles array, then return TRUE.
Return FALSE.

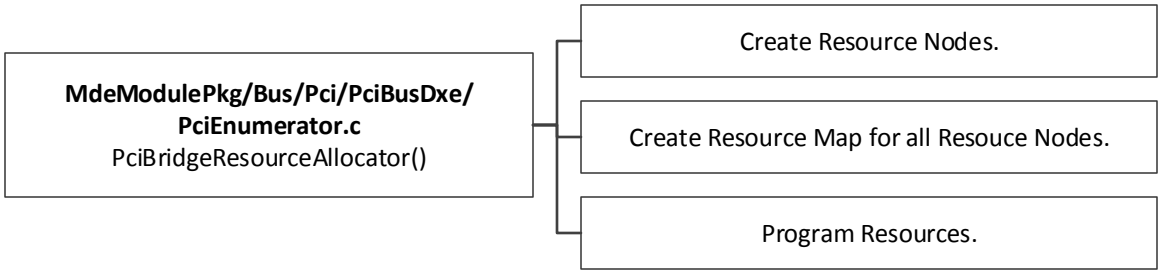

MdeModulePkg/Bus/Pci/PciBusDxe/
PciEnumerator.c
PciEnumerator()
If gFullEnumeration is FALSE, then return PciEnumeratorLight() value.
See PciEnumeratorLight
Open protocol gEfiPciRootBridgeIoProtocolGuid on the specified controller.
Open protocol gEfiPciHostBridgeResourceAllocationProtocolGuid on the Host Bridge handle.
Call NotifyPhase() with EfiPciHostBridgeBeginEnumeration parameter.
See NotifyPhase
Call PciHostBridgeEnumerator().
See PciHostBridgeEnumerator
Call PciHostBridgeResourceAllocator().
See PciHostBridgeResourceAllocator
Call NotifyPhase() with EfiPciHostBridgeEndEnumeration parameter.
See NotifyPhase
Call PciHostBridgeP2CProcess().
See PciHostBridgeP2CProcess
Call PciHostBridgeDeviceAttribute().
See PciHostBridgeDeviceAttribute
Install gEfiPciEnumerationCompleteProtocolGuid protocol on Host Bridge handle.

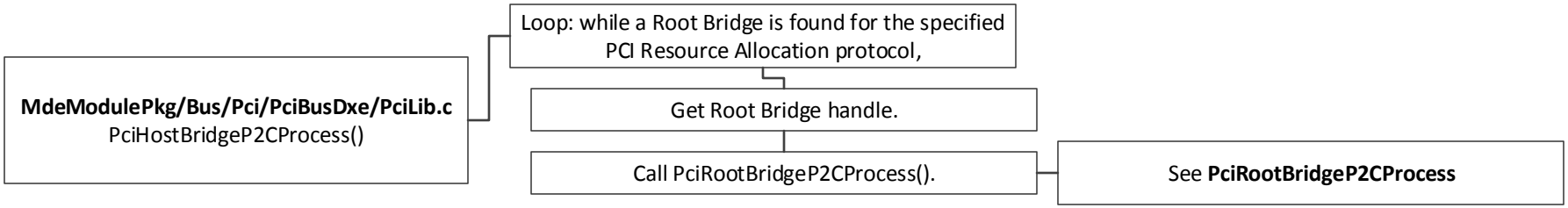

MdeModulePkg/Bus/Pci/PciBusDxe/PciLib.c
PciHostBridgeEnumerator()
Call NotifyPhase() with EfiPciHostBridgeBeginBusAllocation parameter.
See NotifyPhase
Loop: while a Root Bridge is found on the Root Bridge handle,
Create a Root Bridge Device on the handle.
Call PciRootBridgeEnumerator() on the Root Bridge Device.
See PciRootBridgeEnumerator
Call NotifyPhase() with EfiPciHostBridgeEndBusAllocation parameter.
See NotifyPhase
If gPciHotPlugInit != NULL and PcdPciBusHotplugDeviceSupport = TRUE,
Loop: while a Root Bridge is found on the Root Bridge handle,
Call StartBusEnumeration() to get bus information.
Call ResetAllPpbBusNumber() on the Root Bridge Device.
Call AllRootHPCInitialized() to wait for all Hot Plug Controllers to be initialized.
Begin Bus Enumeration for a 2nd time and call code from NotifyPhase() to NotifyPhase() above.
Call NotifyPhase() with EfiPciHostBridgeBeginResourceAllocation parameter.
Loop: while a Root Bridge is found on the Root Bridge handle,
Create a Root Bridge Device on the handle.
Start managing the Root Bridge, get the Bus Range, determine the Root Bridge attributes.
Call PciPciDeviceInfoCollector().
See PciPciDeviceInfoCollector
Call AddHostBridgeEnumerator() to record the Host Bridge handle.

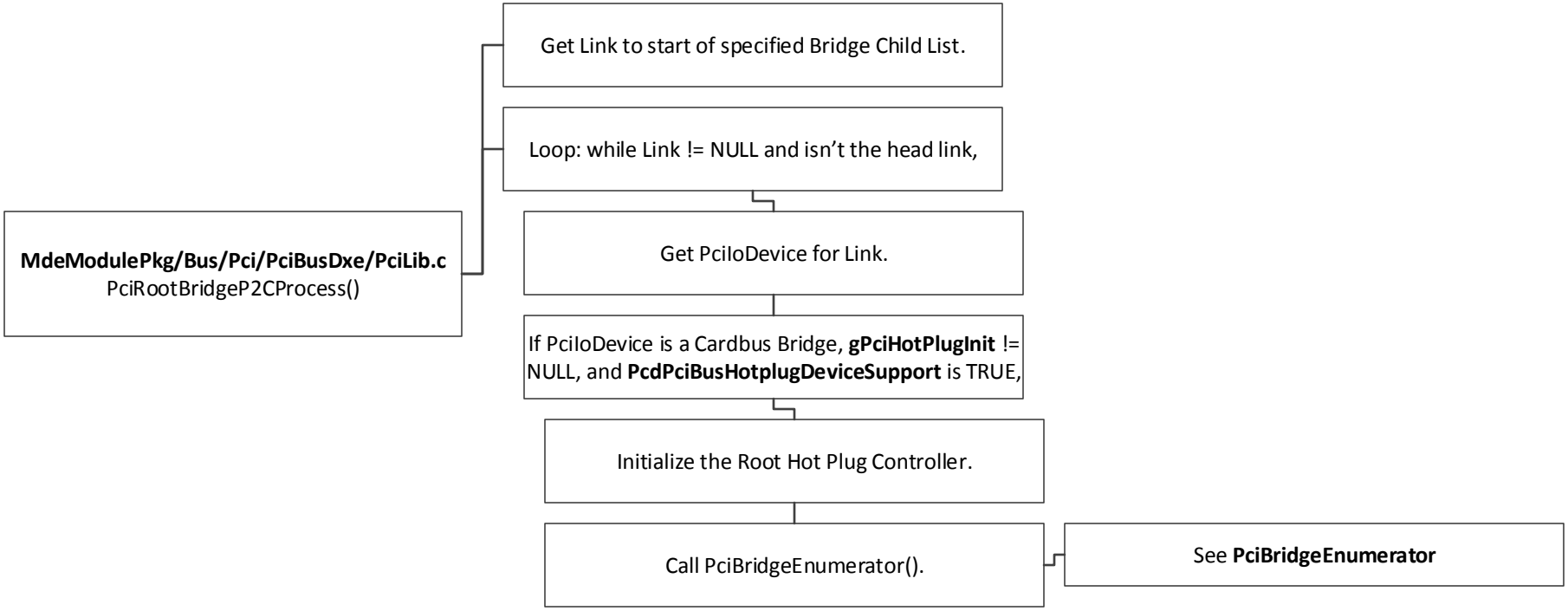

MdeModulePkg/Bus/Pci/PciBusDxe/
PciEnumerator.c
PciRootBridgeEnumerator()
Call StartBusEnumeration() with the specified Resource Allocation protocol to get Configuration Descriptors.
Sort the Configuration Descriptors in ascending order.
Set the Start Bus Number from the first Configuration Descriptor.
Call PciScanBus() to assign a bus number to the specified Root Bridge Device.
Find the bus range which contains the highest bus number.
Modify Configuration Descriptors to mark the End Desctiptor.

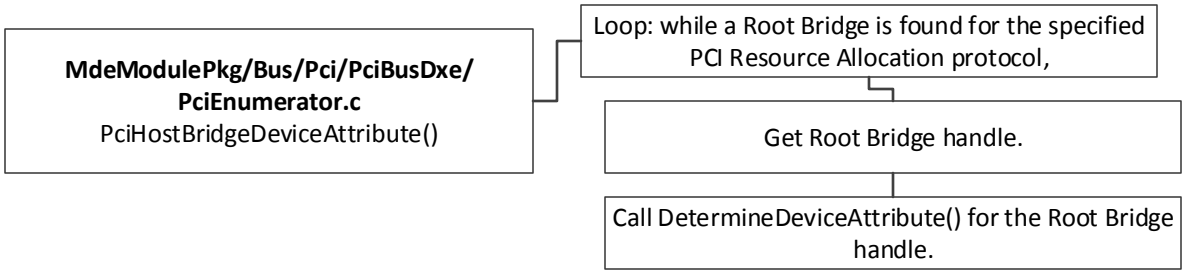

MdeModulePkg/Bus/Pci/PciBusDxe/ PciEnumerator.c
PciBridgeEnumerator()
Call Pci.Read() to get StartBusNumber from Configuration Header offset 0x19.
Call PciAssignBusNumber() to assign StartBusNumber to the specified Bridge.
Call PciPciDeviceInfoCollector().
See **PciPciDeviceInfoCollector**
Call PciBridgeResourceAllocator().
See **PciBridgeResourceAllocator**
Call DetermineDeviceAttribute().

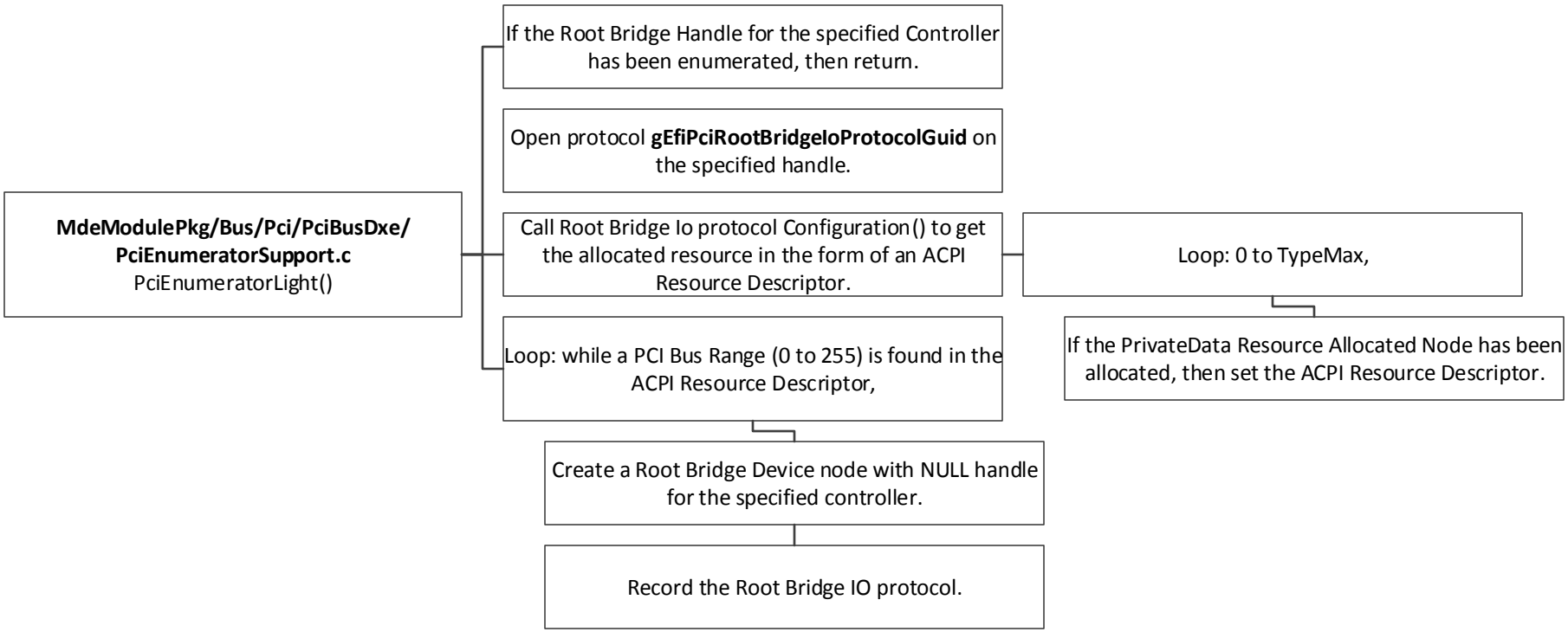

MdeModulePkg/Bus/Pci/PciBusDxe/PciLib.c
PciHostBridgeResourceAllocator()
Loop: while TRUE,
Loop: while a Root Bridge is found for the specified PCI Resource Allocation protocol,
Get Root Bridge handle.
Create Resource Nodes.
Create Resource Map for all Resouce Nodes.
Construct ACPI Resource Requestor.
Call NotifyPhase() with EfiPciHostBridgeAllocateResources parameter.
See NotifyPhase
If PCI Hot Plug is supported, and resource allocation fails,
Loop: while a Root Bridge is found for the specified PCI Resource Allocation protocol,
Get ACPI Resource Node and adjust resource allocation policy for each Root Bridge.
Adjust resource allocation policy for each Host Bridge.
Call NotifyPhase() with EfiPciHostBridgeFreeResources parameter.
See NotifyPhase
Call NotifyPhase() with EfiPciHostBridgeSetResources parameter.
See NotifyPhase
Loop: while a Root Bridge is found for the specified PCI Resource Allocation protocol,
Find Resource Nodes and program the Resources.
Call NotifyPhase() with EfiPciHostBridgeEndResourceAllocation parameter.
See NotifyPhase

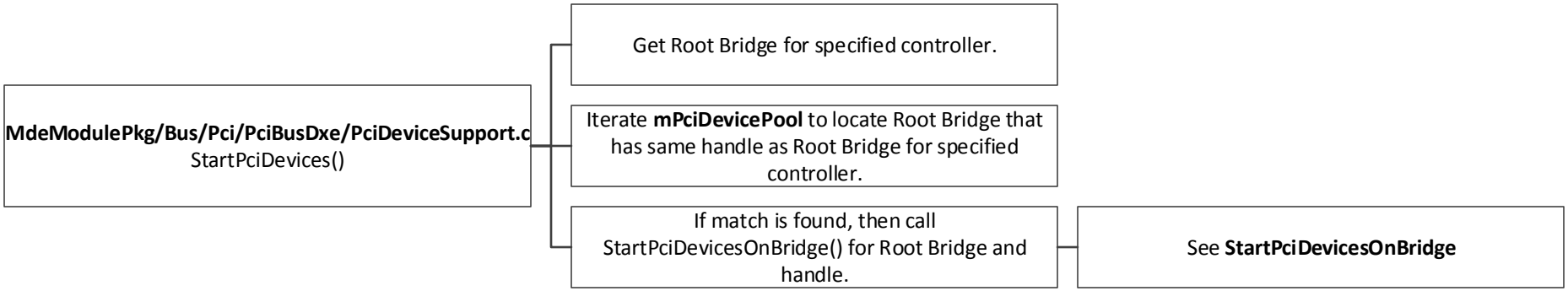

MdeModulePkg/Bus/Pci/PciBusDxe/
PciEnumerator.c
PciBridgeResourceAllocator()
Create Resource Nodes.
Create Resource Map for all Resouce Nodes.
Program Resources.

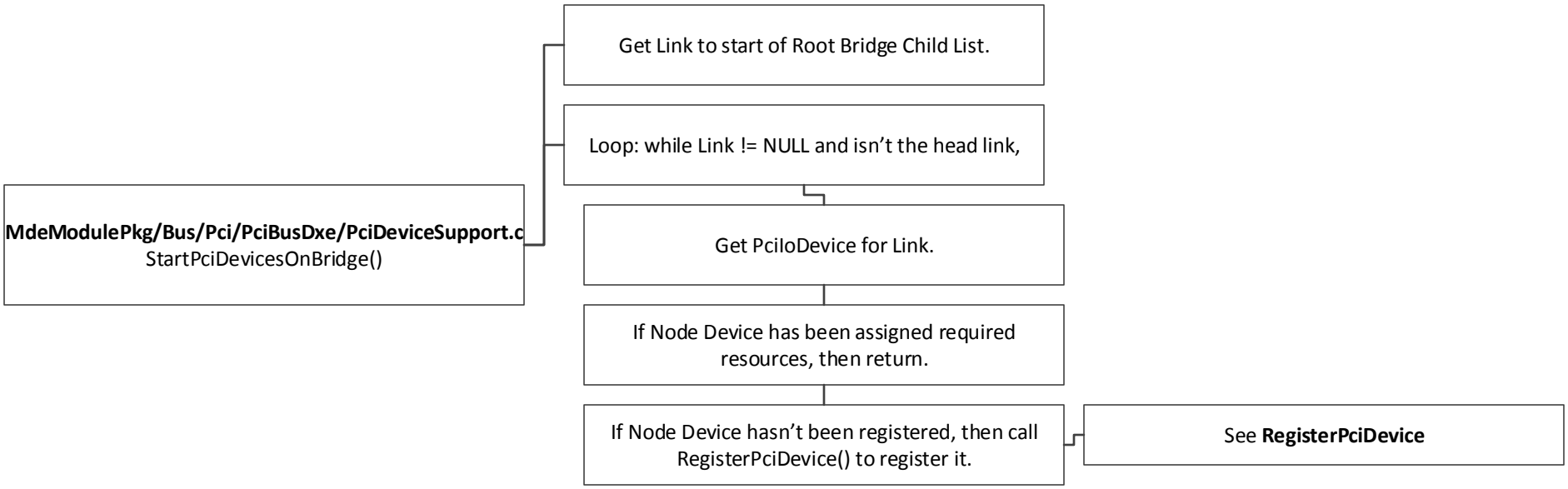

MdeModulePkg/Bus/Pci/PciBusDxe/PciLib.c
PciHostBridgeP2CProcess()
Loop: while a Root Bridge is found for the specified PCI Resource Allocation protocol,
Get Root Bridge handle.
Call PciRootBridgeP2CProcess().
See PciRootBridgeP2CProcess

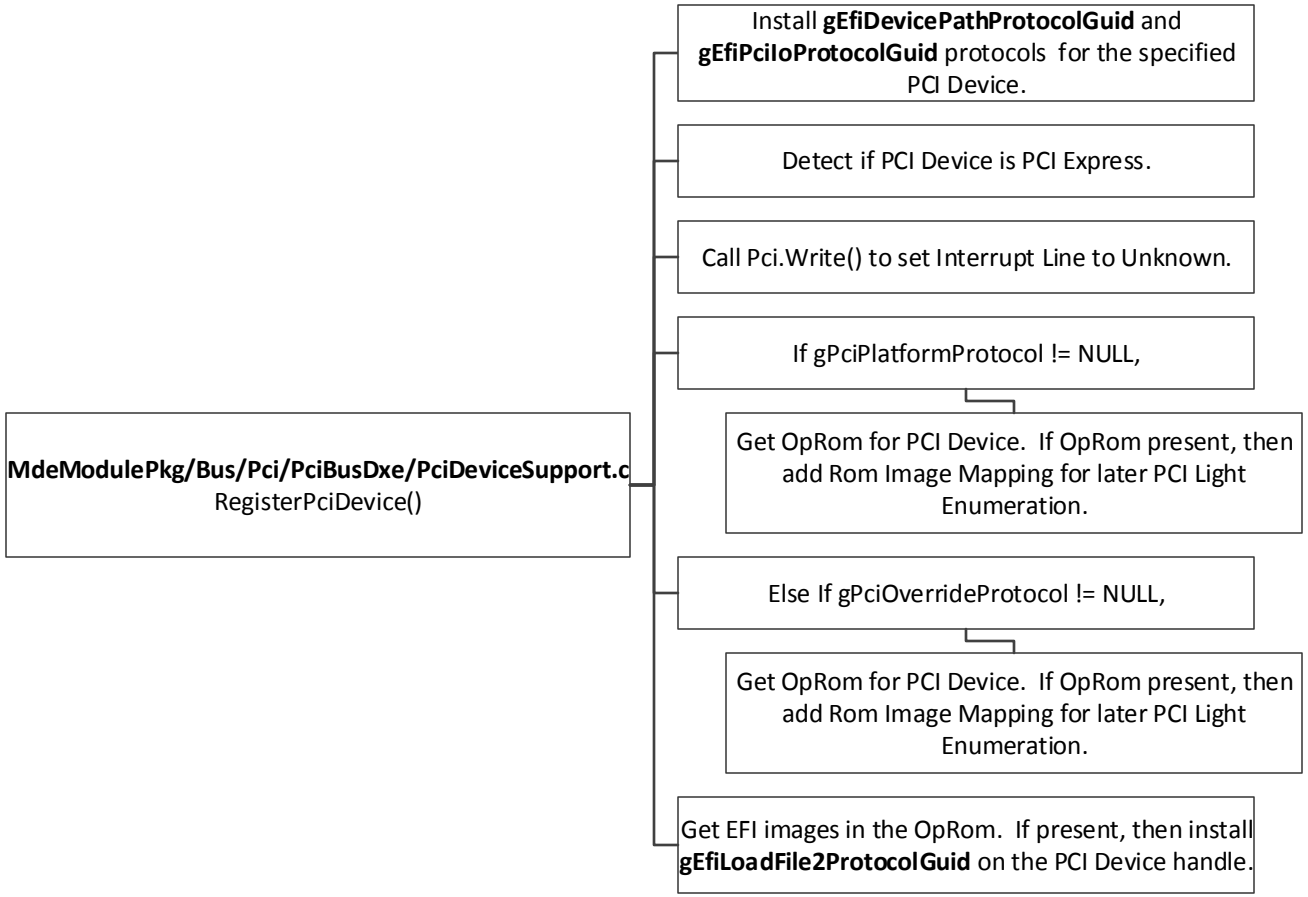

Get Link to start of specified Bridge Child List.
Loop: while Link != NULL and isn't the head link,
MdeModulePkg/Bus/Pci/PciBusDxe/PciLib.c
PciRootBridgeP2CProcess()
Get PciIoDevice for Link.
If PciIoDevice is a Cardbus Bridge, gPciHotPlugInit != NULL, and PcdPciBusHotplugDeviceSupport is TRUE,
Initialize the Root Hot Plug Controller.
Call PciBridgeEnumerator().
See PciBridgeEnumerator

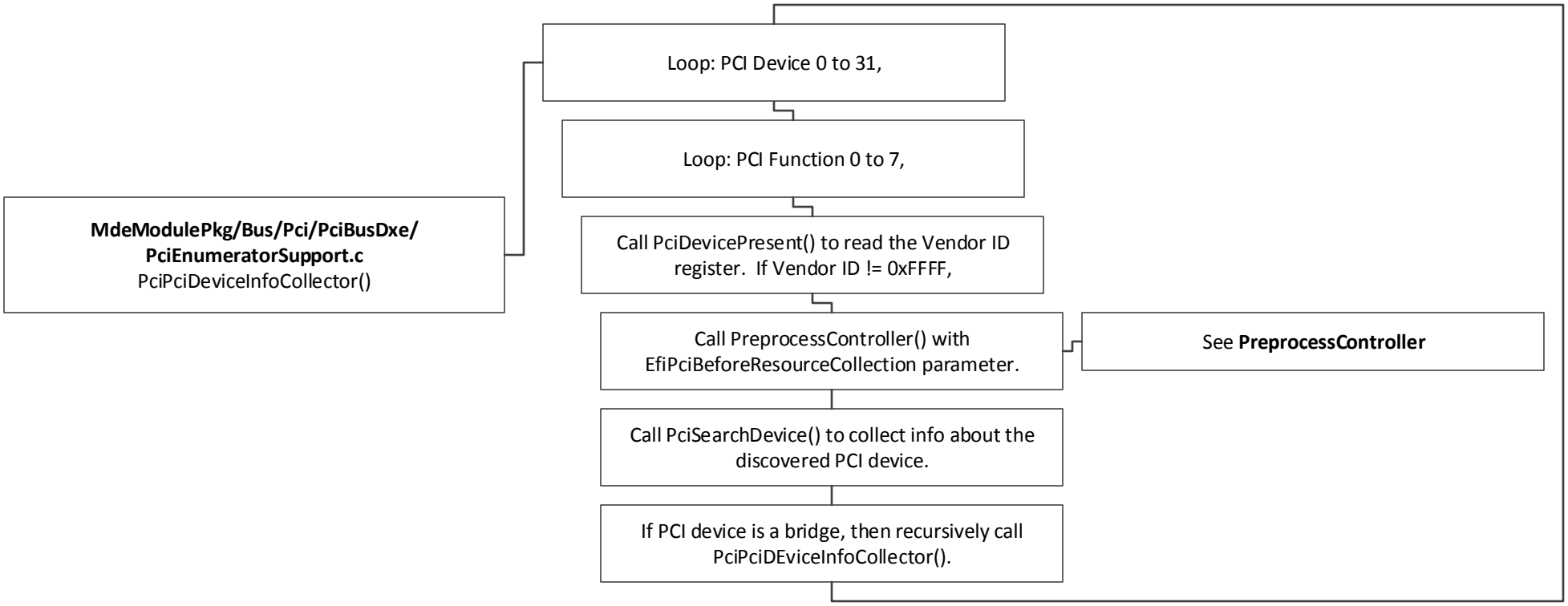

MdeModulePkg/Bus/Pci/PciBusDxe/
PciEnumerator.c
PciHostBridgeDeviceAttribute()
Loop: while a Root Bridge is found for the specified PCI Resource Allocation protocol,
Get Root Bridge handle.
Call DetermineDeviceAttribute() for the Root Bridge handle.

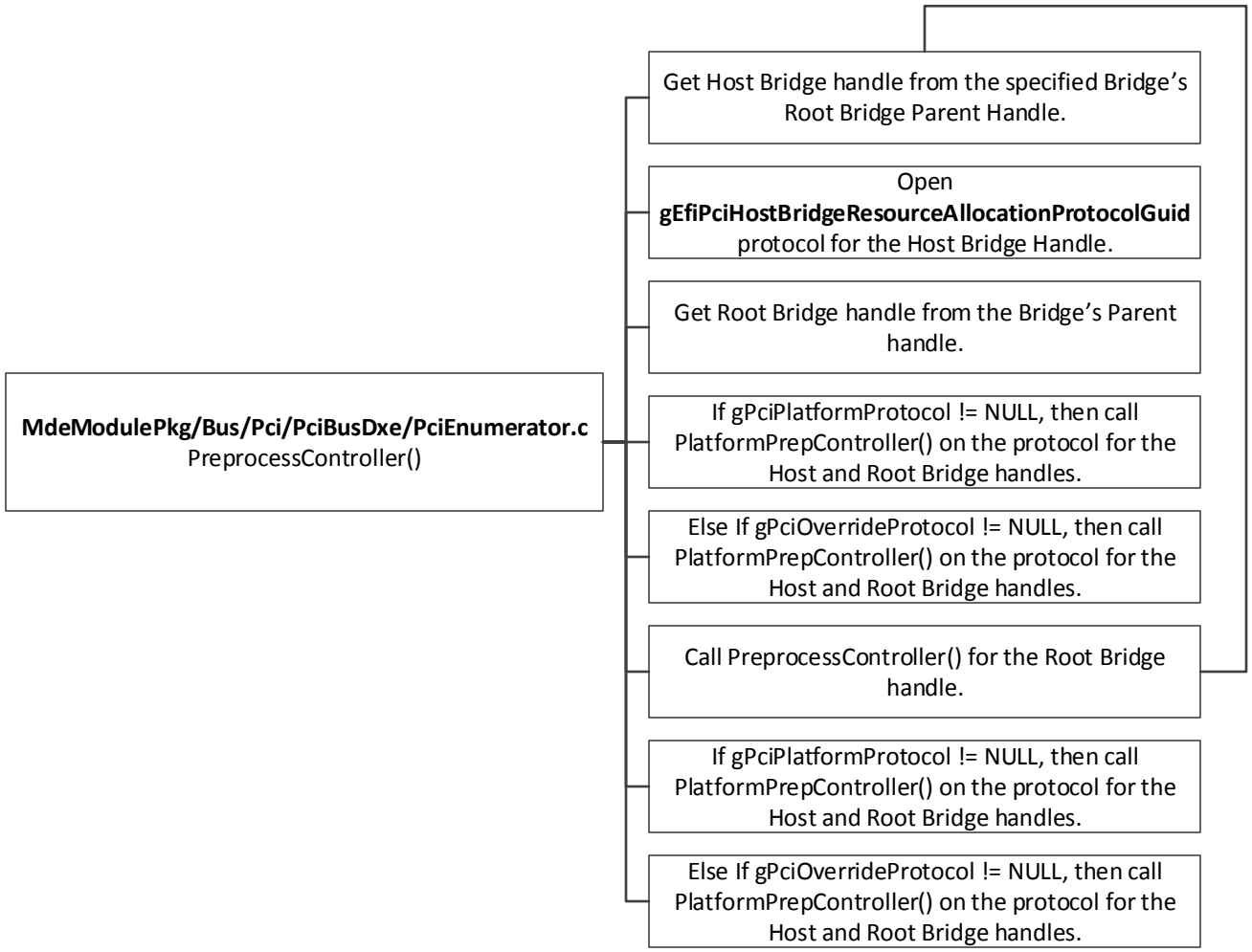

MdeModulePkg/Bus/Pci/PciBusDxe/ PciEnumeratorSupport.c
PciEnumeratorLight()
If the Root Bridge Handle for the specified Controller has been enumerated, then return.
Open protocol **gEfiPciRootBridgeIoProtocolGuid** on the specified handle.
Call Root Bridge Io protocol Configuration() to get the allocated resource in the form of an ACPI Resource Descriptor.
Loop: 0 to TypeMax,
If the PrivateData Resource Allocated Node has been allocated, then set the ACPI Resource Descriptor.
Loop: while a PCI Bus Range (0 to 255) is found in the ACPI Resource Descriptor,
Create a Root Bridge Device node with NULL handle for the specified controller.
Record the Root Bridge IO protocol.


MdeModulePkg/Bus/Pci/PciBusDxe/PciDeviceSupport.c
StartPciDevices()
Get Root Bridge for specified controller.
Iterate **mPciDevicePool** to locate Root Bridge that has same handle as Root Bridge for specified controller.
If match is found, then call StartPciDevicesOnBridge() for Root Bridge and handle.
See **StartPciDevicesOnBridge**


MdeModulePkg/Bus/Pci/PciBusDxe/PciDeviceSupport.c
StartPciDevicesOnBridge()
Get Link to start of Root Bridge Child List.
Loop: while Link != NULL and isn't the head link,
Get PciIoDevice for Link.
If Node Device has been assigned required resources, then return.
If Node Device hasn't been registered, then call RegisterPciDevice() to register it.
See **RegisterPciDevice**


MdeModulePkg/Bus/Pci/PciBusDxe/PciDeviceSupport.c
RegisterPciDevice()
Install gEfiDevicePathProtocolGuid and gEfiPciIoProtocolGuid protocols for the specified PCI Device.
Detect if PCI Device is PCI Express.
Call Pci.Write() to set Interrupt Line to Unknown.
If gPciPlatformProtocol != NULL,
Get OpRom for PCI Device. If OpRom present, then add Rom Image Mapping for later PCI Light Enumeration.
Else If gPciOverrideProtocol != NULL,
Get OpRom for PCI Device. If OpRom present, then add Rom Image Mapping for later PCI Light Enumeration.
Get EFI images in the OpRom. If present, then install gEfiLoadFile2ProtocolGuid on the PCI Device handle.


MdeModulePkg/Bus/Pci/PciBusDxe/
PciEnumeratorSupport.c
PciPciDeviceInfoCollector()
Loop: PCI Device 0 to 31,
Loop: PCI Function 0 to 7,
Call PciDevicePresent() to read the Vendor ID register. If Vendor ID != 0xFFFF,
Call PreprocessController() with EfiPciBeforeResourceCollection parameter.
See PreprocessController
Call PciSearchDevice() to collect info about the discovered PCI device.
If PCI device is a bridge, then recursively call PciPciDEviceInfoCollector().


MdeModulePkg/Bus/Pci/PciBusDxe/PciEnumerator.c
PreprocessController()
Get Host Bridge handle from the specified Bridge's Root Bridge Parent Handle.
Open gEfiPciHostBridgeResourceAllocationProtocolGuid protocol for the Host Bridge Handle.
Get Root Bridge handle from the Bridge's Parent handle.
If gPciPlatformProtocol != NULL, then call PlatformPrepController() on the protocol for the Host and Root Bridge handles.
Else If gPciOverrideProtocol != NULL, then call PlatformPrepController() on the protocol for the Host and Root Bridge handles.
Call PreprocessController() for the Root Bridge handle.
If gPciPlatformProtocol != NULL, then call PlatformPrepController() on the protocol for the Host and Root Bridge handles.
Else If gPciOverrideProtocol != NULL, then call PlatformPrepController() on the protocol for the Host and Root Bridge handles.