The entry point of a **PCIe driver in Linux** depends on whether it's a **PCI device driver** or a **subsystem driver** (e.g., PCIe controller driver for an SoC).

**1. Entry Point for a PCI Device Driver**

A standard **PCI device driver** in Linux follows the **Linux Device Driver Model**. The key entry points are:

**a. Module Initialization (module\_init)**

When the PCI driver is loaded, the kernel calls the function registered with module\_init().

static int \_\_init my\_pci\_driver\_init(void)

{

return pci\_register\_driver(&my\_pci\_driver);

}

module\_init(my\_pci\_driver\_init);

**b. Registering the PCI Driver (pci\_register\_driver)**

The pci\_register\_driver() function registers the driver with the PCI subsystem.

static struct pci\_driver my\_pci\_driver = {

.name = "my\_pci\_device",

.id\_table = my\_pci\_ids, // PCI device IDs this driver supports

.probe = my\_pci\_probe, // Called when a matching PCI device is found

.remove = my\_pci\_remove, // Called when the device is removed

};

**c. Probe Function (probe) – The Main Entry for a Specific Device**

When the PCI subsystem detects a matching device, it calls probe().

static int my\_pci\_probe(struct pci\_dev \*pdev, const struct pci\_device\_id \*id)

{

int ret;

// Enable the PCI device

ret = pci\_enable\_device(pdev);

if (ret) {

dev\_err(&pdev->dev, "Failed to enable PCI device\n");

return ret;

}

// Request PCI regions

ret = pci\_request\_regions(pdev, "my\_pci\_device");

if (ret) {

pci\_disable\_device(pdev);

return ret;

}

// Map BAR (Base Address Register)

void \_\_iomem \*mmio\_base = pci\_iomap(pdev, 0, 0);

if (!mmio\_base) {

pci\_release\_regions(pdev);

pci\_disable\_device(pdev);

return -ENOMEM;

}

// Store private data and initialize the device

pci\_set\_drvdata(pdev, mmio\_base);

dev\_info(&pdev->dev, "PCI device initialized\n");

return 0;

}

**d. Remove Function (remove) – Cleanup on Device Removal**

When the device is removed, the remove() function is called.

static void my\_pci\_remove(struct pci\_dev \*pdev)

{

void \_\_iomem \*mmio\_base = pci\_get\_drvdata(pdev);

if (mmio\_base)

pci\_iounmap(pdev, mmio\_base);

pci\_release\_regions(pdev);

pci\_disable\_device(pdev);

dev\_info(&pdev->dev, "PCI device removed\n");

}

**2. Entry Point for PCIe Host/Controller Driver**

If you're working on a **PCIe root complex driver** (for an SoC), the entry point is usually:

* **pci\_host\_probe()**: Probes PCIe root complex
* **platform\_driver\_register()**: Registers the PCIe controller driver

Example:

static int my\_pcie\_probe(struct platform\_device \*pdev)

{

struct device \*dev = &pdev->dev;

// Initialize PCIe host controller

struct pci\_host\_bridge \*bridge = devm\_pci\_alloc\_host\_bridge(dev, 0);

if (!bridge)

return -ENOMEM;

return pci\_host\_probe(bridge);

}

**Summary**

* **For PCIe device drivers**, entry points are:
  + module\_init() → pci\_register\_driver() → probe()
* **For PCIe host controllers**, entry points are:
  + platform\_driver\_register() → pci\_host\_probe()

Would you like more details on specific parts like **BAR mapping**, **MSI interrupts**, or **config space access**?

Top of Form