MTCA
PCI Express and PCI Express Hot Plug

L. Petrosyan
PCI Express (Peripheral Component Interconnect Express) is a computer bus which moves information between the internal Hardware of a computer system (including the CPU and RAM) and peripheral devices. It is a collection of wires and protocols that allows for the expansion of a computer.

- PCI Express is a serial point-to-point connection.
- Each device sits on its own dedicated bus, which in PCIe lingo is called **Link**.
- On one bus (**Link**) there can be only two devices.
- Each link is composed of one or more **Lanes**.
- Each Lane is a **differential signal pair** in each direction.
PCI Express Port

The **PCI express Device** is identified by **Vendor/Device IDs** are commonly called the **PCI ID**. The 16-bit vendor ID is allocated by the PCI-SIG. The 16-bit device ID is then assigned by the vendor. The **PCI express Device** addressed by **Bus number** or/and by **Memory address** assigned to the current device.

- Port is the interface between a PCI Express component and the Link
- It consists of differential transmitters and receivers

- The number of Link lines and transmitters of the port can differ
PCI Express Switch

- PCI Express is a point-to-point connection between two devices
- To add more devices
- we need more buses
- PCI Express Switch add buses and controls several point-to-point serial connections.

- Switch has two or more logical PCI-to-PCI bridges, each bridge associated with a switch port
The **Root Complex** denotes the device that connects the CPU and memory subsystems to the PCI Express fabric.

- It may support one or more PCI Express Ports.

- Root Complex generates PCI Express configuration and enumerates the System.
- PCI Express transactions use the address (bus:device) and memory routing.
- The Root Complex Bus number initialize to 0.
• multiple switch devices can be connected to ports on the root complex or cascaded
PCI Express and MTCA

Back Plane

- MCH in MCH Slot
- CPU in Upstream Slot
- AMC Modules

MCH

PCle Switch

Root Compl

Port

CPU

Memory

AMC

AMC
PCI Express and MTCA

- MCH sets Up Upstream Port of the PCIe Switch
PCI Express and MTCA

- MCH sets Up Upstream Port of the PCIe Switch
- MCH connects links to AMC slots
PCI Express and MTCA

- MCH sets Up Upstream Port of the PCIe Switch
- MCH connects links to AMC slots
- Root Complex configures and enumerates the PCI Express Busses
- OS provided PCI Express Bus Driver crates system Devices for all PCI Express devices and allocates memories
  - At this point all devices visible in `lspci`
  - The user application could uses system Device Files to map Device memory for accessing to the device

```
sys/bus/pci/devices/xxx
```
PCI Express and MTCA

- MCH sets Up Upstream Port of the PCIe Switch
- MCH connects links to AMC slots
- Root Complex configures and enumerates the PCI Express Busses
- OS provided PCI Express Bus Driver crates system Devices for all PCI Express devices and allocates memories
  - At this point all devices visible in `lspci`
  - The user application could uses system Device Files to map Device memory for accessing to the device
- RCI Express Bus Driver call for every Device appropriate driver, according of PCI Vendor/Device IDs
- Device Driver maps Device memories and crates Device File as entry piont
  - *The user application uses Device File for accessing to the Device by means of File Operation functions*
PCI Express and MTCA

Does `lspci` list the device

- YES
- NO

Does `lspci` list MCH Switch

- YES
- NO

Is the AMC in M4 state

- NO
- YES

Try to reboot MCH, call expert

- YES
- NO

Is the AMC links are connected

- NO
- YES

Try to reboot AMC, call expert

- YES
- NO

Try to reboot CPU, call expert

- YES
- NO

Is the device file in `/dev`

- NO
- YES

Check the kernel log file and call expert

- GOOD
-

Is the device BARs enabled

- NO
- YES

Is the device driver loaded

- NO
- YES

Install the device driver

- YES
- NO

Is the another driver for the same ID loaded

- YES
- NO

Unload the driver

- YES
- NO

Is the device driver installed

- NO
- YES

Is the another driver for the same ID loaded

- YES
- NO

Unload the driver

- YES
- NO

Is the device BARs enabled

- NO
- YES

Install the device driver

- YES
- NO

Is the device driver installed

- NO
- YES

Check the kernel log file and call expert

- GOOD
-
PCI Express and MTCA

Lspci –H1

00:00.0 Host bridge: Intel Corporation 2nd Generation Core Processor Family DRAM Controller (rev 09)
00:01.0 PCI bridge: Intel Corporation Xeon E3-1200/2nd Generation Core Processor Family PCI Express Root Port (rev 09)
00:01.1 PCI bridge: Intel Corporation Xeon E3-1200/2nd Generation Core Processor Family PCI Express Root Port (rev 09)
00:02.0 VGA compatible controller: Intel Corporation 2nd Generation Core Processor Family Integrated Graphics Controller (rev 09)
00:19.0 Ethernet controller: Intel Corporation 82579LM Gigabit Network Connection (rev 04)
00:1a.0 USB controller: Intel Corporation 6 Series/C200 Series Chipset Family USB Enhanced Host Controller #2 (rev 04)
00:1c.0 PCI bridge: Intel Corporation 6 Series/C200 Series Chipset Family PCI Express Root Port 1 (rev b4)
00:1d.0 USB controller: Intel Corporation 6 Series/C200 Series Chipset Family USB Enhanced Host Controller #1 (rev 04)
00:1f.0 ISA bridge: Intel Corporation 6 Series/C200 Series Chipset Family LPC Controller (rev 04)
00:1f.4 SMBus: Intel Corporation 6 Series/C200 Series Chipset Family SMBus Controller (rev 04)
00:1f.5 IDE interface: Intel Corporation 6 Series/C200 Series Chipset Family 2 port SATA IDE Controller (rev 04)
01:00.0 PCI bridge: Integrated Device Technology, Inc. [IDT] Device 808f
01:00.2 System peripheral: Integrated Device Technology, Inc. [IDT] Device 808f
02:08.0 PCI bridge: Integrated Device Technology, Inc. [IDT] Device 808f
02:0c.0 PCI bridge: Integrated Device Technology, Inc. [IDT] Device 808f
03:00.0 PCI bridge: PLX Technology, Inc. Device 8748 (rev ba)
04:00.0 PCI bridge: PLX Technology, Inc. Device 8748 (rev ba)
04:01.0 PCI bridge: PLX Technology, Inc. Device 8748 (rev ba)
04:02.0 PCI bridge: PLX Technology, Inc. Device 8748 (rev ba)
04:08.0 PCI bridge: PLX Technology, Inc. Device 8748 (rev ba)
04:09.0 PCI bridge: PLX Technology, Inc. Device 8748 (rev ba)
04:0a.0 PCI bridge: PLX Technology, Inc. Device 8748 (rev ba)
04:0b.0 PCI bridge: PLX Technology, Inc. Device 8748 (rev ba)
04:10.0 PCI bridge: PLX Technology, Inc. Device 8748 (rev ba)
04:11.0 PCI bridge: PLX Technology, Inc. Device 8748 (rev ba)
04:12.0 PCI bridge: PLX Technology, Inc. Device 8748 (rev ba)
04:13.0 PCI bridge: PLX Technology, Inc. Device 8748 (rev ba)
07:00.0 Communication synchronizer: Xilinx Corporation Device 0020
0a:00.0 Signal processing controller: Xilinx Corporation Device 0088
12:00.0 Ethernet controller: Intel Corporation 82580 Gigabit Backplane Connection (rev 01)
12:00.1 Ethernet controller: Intel Corporation 82580 Gigabit Backplane Connection (rev 01)

SIS8300 board
Vendor ID 10EE (XILINX)
Device ID 0088 (DESY dev. ID)
PCI Express and MTCA

Lspci –H1

00:00.0 Host bridge: Intel Corporation 2nd Generation Core Processor Family DRAM Controller (rev 09)
00:01.0 PCI bridge: Intel Corporation Xeon E3-1200/2nd Generation Core Processor Family PCI Express Root Port (rev 09)
00:01.1 PCI bridge: Intel Corporation Xeon E3-1200/2nd Generation Core Processor Family PCI Express Root Port (rev 09)
00:02.0 VGA compatible controller: Intel Corporation 2nd Generation Core Processor Family DRAM Controller (rev 09)
00:19.0 Ethernet controller: Intel Corporation 82579LM Gigabit Network Connection (rev 04)
00:1a.0 USB controller: Intel Corporation 2nd Generation Core Processor Family PCI Express Root Port 1 (rev b4)
00:1d.0 USB controller: Intel Corporation 2nd Generation Core Processor Family PCI Express Root Port 1 (rev 04)
00:1e.0 USB controller: Intel Corporation 2nd Generation Core Processor Family PCI Express Root Port 1 (rev 04)

Lspci –H1 -t

00:00.0 Host bridge: Intel Corporation 2nd Generation Core Processor Family DRAM Controller (rev 09)
00:01.0 PCI bridge: Intel Corporation Xeon E3-1200/2nd Generation Core Processor Family PCI Express Root Port (rev 09)
00:01.1 PCI bridge: Intel Corporation Xeon E3-1200/2nd Generation Core Processor Family PCI Express Root Port (rev 09)
00:02.0 VGA compatible controller: Intel Corporation 2nd Generation Core Processor Family DRAM Controller (rev 09)
00:19.0 Ethernet controller: Intel Corporation 82579LM Gigabit Network Connection (rev 04)
00:1a.0 USB controller: Intel Corporation 2nd Generation Core Processor Family PCI Express Root Port 1 (rev b4)
00:1d.0 USB controller: Intel Corporation 2nd Generation Core Processor Family PCI Express Root Port 1 (rev 04)
00:1e.0 USB controller: Intel Corporation 2nd Generation Core Processor Family PCI Express Root Port 1 (rev 04)

00:04.0 Signal processing controller: Xilinx Corporation Device 0020
0a:00.0 Signal processing controller: Xilinx Corporation Device 0088
10:00.0 Ethernet controller: Intel Corporation 82580 Gigabit Backplane Connection (rev 01)
12:00.1 Ethernet controller: Intel Corporation 82580 Gigabit Backplane Connection (rev 01)
PCI Express and MTCA

Lspci –H1
00:00.0 Host bridge: Intel Corporation 2nd Generation Core Processor Family DRAM Controller (rev 09)
00:01.0 PCI bridge: Intel Corporation Xeon E3-1200/2nd Generation Core Processor Family PCI Express Root Port (rev 09)
00:01.1 PCI bridge: Intel Corporation Xeon E3-1200/2nd Generation Core Processor Family PCI Express Root Port (rev 09)
00:02.0 VGA compatible controller: Intel Corporation 2nd Generation Core Processor Family Integrated Graphics Controller (rev 09)
00:02.0 Ethernet controller: Intel Corporation 82579LM Gigabit Network Connection (rev 04)
00:0a.0 USB controller: Intel Corporation 6 Series/C200 Series Chipset Family USB Enhanced Host Controller #2 (rev 04)
00:1c.0 PCI bridge: Intel Corporation 6 Series/C200 Series Chipset Family PCI Express Root Port 1 (rev b4)
00:1d.0 USB controller: Intel Corporation 6 Series/C200 Series Chipset Family USB Enhanced Host Controller #1 (rev 04)
00:1f.0 ISA bridge: Intel Corporation QM67 Express Chipset Family LPC Controller (rev 04)
00:1f.2 IDE interface: Intel Corporation 6 Series/C200 Series Chipset Family USB Enhanced Host Controller #1 (rev 04)
00:1f.2 IDE interface: Intel Corporation 6 Series/C200 Series Chipset Family USB Enhanced Host Controller #1 (rev 04)
01:00.0 PCI bridge: Integrated Device Technology, Inc. [IDT] Device 808f
01:00.2 System peripheral: Integrated Device Technology, Inc. [IDT] Device 808f
02:08.0 PCI bridge: Integrated Device Technology, Inc. [IDT] Device 808f
02:0c.0 PCI bridge: Integrated Device Technology, Inc. [IDT] Device 808f
03:00.0 PCI bridge: PLX Technology, Inc. Device 8748 (rev ba)
04:00.0 PCI bridge: PLX Technology, Inc. Device 8748 (rev ba)
04:01.0 PCI bridge: PLX Technology, Inc. Device 8748 (rev ba)
04:02.0 PCI bridge: PLX Technology, Inc. Device 8748 (rev ba)
04:08.0 PCI bridge: PLX Technology, Inc. Device 8748 (rev ba)
04:09.0 PCI bridge: PLX Technology, Inc. Device 8748 (rev ba)
04:0a.0 PCI bridge: PLX Technology, Inc. Device 8748 (rev ba)
04:0b.0 PCI bridge: PLX Technology, Inc. Device 8748 (rev ba)
04:10.0 PCI bridge: PLX Technology, Inc. Device 8748 (rev ba)
04:11.0 PCI bridge: PLX Technology, Inc. Device 8748 (rev ba)
04:12.0 PCI bridge: PLX Technology, Inc. Device 8748 (rev ba)
04:13.0 PCI bridge: PLX Technology, Inc. Device 8748 (rev ba)
07:00.0 Communication synchronizer: Xilinx Corporation Device 0020
0a:00.0 System peripheral: Xilinx Corporation Device 0020
0b:00.0 Host bridge: Intel Corporation 2nd Generation Core Processor Family DRAM Controller (rev 09)
0c:00.0 Host bridge: Intel Corporation 2nd Generation Core Processor Family DRAM Controller (rev 09)
0d:00.0 Host bridge: Intel Corporation 2nd Generation Core Processor Family DRAM Controller (rev 09)
0e:00.0 Serial port: Intel Corporation 82580 Gigabit Backplane Connection (rev 01)
0f:00.0 Serial port: Intel Corporation 82580 Gigabit Backplane Connection (rev 01)

Lspci –H1 -vvv -s 0a:00.0
0a:00.0 Signal processing controller: Xilinx Corporation Device 0088
0b:00.0 Signal processing controller: Xilinx Corporation Device 0088

DOOCs
L.Petrosyan MCS4 DESY MicroTCA workshop for industry and research 10/12/2016
16
PCI Express and MicroTCA

Lspci –H1

04:09.0 Host bridge: Intel Corporation 2nd Generation Core Processor Family DRAM Controller (rev 09)
00:00.0 PCI bridge: Intel Corporation Xeon E3-1200/2nd Generation Core Processor Family PCI Express Root Port (rev 09)
00:01.1 PCI bridge: Intel Corporation Xeon E3-1200/2nd Generation Core Processor Family PCI Express Root Port (rev 09)
00:02.0 VGA compatible controller: Intel Corporation 2nd Generation Core Processor Family Integrated Graphics Controller (rev 09)
00:19.0 Ethernet controller: Intel Corporation 82579LM Gigabit Network Connection (rev 04)
00:1a.0 USB controller: Intel Corporation 6 Series/C200 Series Chipset Family USB Enhanced Host Controller #2 (rev 04)
00:1c:0.0 PCI bridge: Intel Corporation 6 Series/C200 Series Chipset Family PCI Express Root Port 1 (rev b4)
00:1d:0.0 USB controller: Intel Corporation 6 Series/C200 Series Chipset Family USB Enhanced Host Controller (rev 04)
00:1f.0 ISA bridge: Intel Corporation QM67 Express Chipset Family LPC Controller (rev 04)
00:1f.2 IDE interface: Intel Corporation 6 Series/C200 Series Chipset Family SATA IDE Controller (rev 04)
00:1f.3 IDE interface: Intel Corporation 6 Series/C200 Series Chipset Family SM Bus Controller (rev 04)
00:1f.1 IDE interface: Intel Corporation 6 Series/C200 Series Chipset Family 2 port SATA IDE Controller (rev 04)
00:1f.0 PCI bridge: Integrated Device Technology, Inc. [IDT] Device 808f
01:00.2 System peripheral: Integrated Device Technology, Inc. [IDT] Device 808f
02:00.0 PCI bridge: Integrated Device Technology, Inc. [IDT] Device 808f
02:0x.0 PCI bridge: Integrated Device Technology, Inc. [IDT] Device 808f
04:00.0 PCI bridge: PLX Technology, Inc. Device 8748 (rev ba)
04:01.0 PCI bridge: PLX Technology, Inc. Device 8748 (rev ba)
04:02.0 PCI bridge: PLX Technology, Inc. Device 8748 (rev ba)
04:08.0 PCI bridge: PLX Technology, Inc. Device 8748 (rev ba)
04:09.0 PCI bridge: PLX Technology, Inc. Device 8748 (rev ba)
04:0a.0 PCI bridge: PLX Technology, Inc. Device 8748 (rev ba)
04:0b.0 PCI bridge: PLX Technology, Inc. Device 8748 (rev ba)
04:0c.0 PCI bridge: PLX Technology, Inc. Device 8748 (rev ba)
04:0d.0 PCI bridge: PLX Technology, Inc. Device 8748 (rev ba)
04:12.0 PCI bridge: PLX Technology, Inc. Device 8748 (rev ba)
04:13.0 PCI bridge: PLX Technology, Inc. Device 8748 (rev ba)
07:00.0 Communication synchronizer: Xilinx Corporation Device 0020

0a:00.0 Signal processing controller: Xilinx Corporation Device 0088
12:00.0 Ethernet controller: Intel Corporation 82580 Gigabit Backplane Connection (rev 01)
12:01.0 Ethernet controller: Intel Corporation 82580 Gigabit Backplane Connection (rev 01)

Lspci –H1 -vvy -s 04:09.0

04:09.0 PCI bridge: PXI Technology, Inc. Device 8748 (rev ba) [prog-if 00 [Normal decode]]

Bus: primary=0, secondary=0, subordinate=0, sec_latency=0

Memory behind bridge: c0000000-cbfffff


Flags: PMEOn= D0- D1- D2- AuxCurrent=0mA- PME(D0+, D1+, D2+, D3hot+, D3cold+)

Status: D0 NoSoftRst+ PME Enable- D5el=0 DScale=0 PME-

Capabilities: [48] MSI: Enable= Count=1/8 Maskable= 64bit+

Address: 00000000-fe000478 Data: 0000

Masking: 0000000f Pending: 00000000

Capabilities: [68] Express (v2) Downstream Port (Slot), MSI 00

DevCap: MaxPayload 512 bytes, PhmFunct= 0, Latency L0s = 64ns, L1 = 1us ExtTag= RBE+ FlrReset-

DevCtl: Report errors: Correctable- Non-Fatal- Fatal- Unsupported- RlxOrd= ExtTag= PhntFunct= AuxPwr= NoSnoop-

MaxPayload 128 bytes, MaxReadReq 128 bytes

DevSta: CorrErr= UnCorrErr- FatalErr- Uns suspReq= AuxPwr= TransPend-

LnkCap: Port #9, Speed unknown, Width x4, ASPM L0s L1, Latency L0 = <4us, L1 = <4us

ClockPM= Surprise+ LlAcqReq= BwNot+ ASPM Disabled; Disabled- Retrain- CommClk-

ExtSynchron- ClockPM= AuxWidDis- BWInt= AutBWInt-

LnkSta: Speed 2.5GT/s, Width x4, TrErr- Train- SlotClk- DLActive+ BWmgmt+ ABWmgmt-

SltCap: AttnBlt+ PwrCtrl+ MRL+ AttnInd+ PwrInd+ HotPlug+ Surprise-

Slot #6, PowerLimit 25.000W; Interlock= NoCompl-

SltCtl: Enable: AttnBlt+ PwrFlt- MRL+ PresDet+ CmcCplt+ HPIRq+ LinkChg-

Control: AttnInd Off, PwrInd On, Power- Interlock-

SltSta: Status: AttnBlt- PowerFlt- MRL- CmcCplt- PresDet- Interlock- Changed: MRL- PresDet- LinkState+

MICRO TCA WORKSHOP FOR INDUSTRY AND RESEARCH

10/12/2016
• Checking MCH Link connections

nat> show_fru

FRU Information:
----------------
FRU  Device  State  Name
==========================================
0   MCH      M4    NMCH-CM
3   mcmc1    M4    NAT-MCH-MCMC
5   AMC1     M4    CCT AM 310/302
6   AMC2     M4    ADB7000
7   AMC3     M4    X2TIMER
8   AMC4     M4    AMC-ADIO24
9   AMC5     M4    DAMC2V2
10  AMC6     M4    SIS8300
11  AMC7     M4    SIS8300
14  AMC10    M4    CCT AM 900/412
40  CU1      M4    Cooling Unit
41  CU2      M4    Cooling Unit
53  PM4      M4    NAT-PM-DC
60  Clk1     M4    MCH-Clock
61  Hub1     M4    MCH-PCIe

nat> show_link_state

AMC  1 Port  0 is Ethernet - 1000Base-BX
AMC  1 Port  4 is PCIe - x4 - 2,5 GT/s
AMC  1 Port  5 is PCIe - x4 - 2,5 GT/s
AMC  1 Port  6 is PCIe - x4 - 2,5 GT/s
AMC  1 Port  7 is PCIe - x4 - 2,5 GT/s
AMC  3 Port  4 is PCIe - x1 - 2,5 GT/s
AMC  4 Port  4 is PCIe - x1 - 2,5 GT/s
AMC  6 Port  4 is PCIe - x4 - 2,5 GT/s
AMC  6 Port  5 is PCIe - x4 - 2,5 GT/s
AMC  6 Port  6 is PCIe - x4 - 2,5 GT/s
AMC  6 Port  7 is PCIe - x4 - 2,5 GT/s
AMC 10 Port  0 is Ethernet - 1000Base-BX
AMC 10 Port  4 is PCIe - x4 - 8,0 GT/s
AMC 10 Port  5 is PCIe - x4 - 8,0 GT/s
AMC 10 Port  6 is PCIe - x4 - 8,0 GT/s
AMC 10 Port  7 is PCIe - x4 - 8,0 GT/s
We could see the MCH PCIe Switch but not our Device
The problem is here
1. Check is the Device powered ON
2. Check Link_state in MCH
3. Check Kernel log file for any PCIe errors
We could not see the MCH PCIe Switch
The problem is here
1. Check is the CPU in Upstream Slot
2. Try to reboot the MCH and CPU
Lspci –H1
00:00.0 Host bridge: Intel Corporation 2nd Generation Core Processor Family DRAM Controller (rev 09)
00:01.0 PCI bridge: Intel Corporation Xeon E3-1200/2nd Generation Core Processor Family PCI Express Root Port (rev 09)
02:08.0 PCI bridge: Integrated Device Technology, Inc. [IDT] Device 808f
02:0c.0 PCI bridge: Integrated Device Technology, Inc. [IDT] Device 808f
03:00.0 PCI bridge: PLX Technology, Inc. Device 8748 (rev ba)
04:00.0 PCI bridge: PLX Technology, Inc. Device 8748 (rev ba)
04:01.0 PCI bridge: PLX Technology, Inc. Device 8748 (rev ba)
04:02.0 PCI bridge: PLX Technology, Inc. Device 8748 (rev ba)
04:08.0 PCI bridge: PLX Technology, Inc. Device 8748 (rev ba)
04:09.0 PCI bridge: PLX Technology, Inc. Device 8748 (rev ba)
04:0a.0 PCI bridge: PLX Technology, Inc. Device 8748 (rev ba)
04:0b.0 PCI bridge: PLX Technology, Inc. Device 8748 (rev ba)
04:10.0 PCI bridge: PLX Technology, Inc. Device 8748 (rev ba)
04:11.0 PCI bridge: PLX Technology, Inc. Device 8748 (rev ba)
04:12.0 PCI bridge: PLX Technology, Inc. Device 8748 (rev ba)
04:13.0 PCI bridge: PLX Technology, Inc. Device 8748 (rev ba)
0a:00.0 Signal processing controller: Xilinx Corporation Device 0088
07:00.0 Communication synchronizer: Xilinx Corporation Device 0020
12:00.0 Ethernet controller: Intel Corporation 82580 Gigabit Backplane Connection (rev 01)
12:00.1 Ethernet controller: Intel Corporation 82580 Gigabit Backplane Connection (rev 01)

AMC 1
Vendor ID 10EE (XILINX)
Device ID 0088 (DESY dev. ID)

The Device is on the PCIe Bus
Check is the Device driver binded to the Device
1. Check lspci –H1 –vvv
2. Check Decvice Driver File in /dev
PCI Express Hot Plug

- A Method of replacing failed expansions cards keeping the OS and other services running during the repair
- Shutting down and restarting software associated with the failed device

To accomplish those task the Hot Plug has to:
- Monitor of the PCI Express slot events and reports these events to software via interrupts
- Selectively turns ON and OFF the Power and Attention Indicators
- Prepares the Card, Slot and processes for the card’s removal or insertion
- Remove or apply power to the Card connector
**Hardware Components**

PCI Express Switch Hot Plug **Hot Plug Controller**
- Receives and processes commands Issued by the Hot Plug Software components and Port interface
- One controller for each Root or Switch port

PCI Express Port Interface
- Controls Port components: power, Indicators and Switches

**Software Components**

User Interface
- Permits the end user to control and monitor Hot Plug

Hot Plug Services
- A OS provided software components that processes Hot Plug requests issued by User and Hardware

Hot Plug System Driver
- Controls the Hot Plug Controller

Device Driver
- Prepares the Device to be removed or initialise the Device after insertion

---

For performance above the stated conditions some new components step on the stage

![Diagram of PCI Express Hot Plug components](image)
PCI Express Hot Plug and MTCA
PCI Express Hot Plug and MTCA

AMC
- MMC
  - Hot Swap Open
  - Blue LED Blink
  - Quiesce

MCH
- MCMC
  - Hot Plug Controller
  - IPMI Interrupt
  - Check Slot Status
  - Set Slot Control Power OFF

OS
- Hot Plug Driver
- Hot Plug Service
- Device Driver
  - Call Device Driver Remove
  - Back from Device Driver Remove

udev

Remove Payload
Red LED ON
Green LED OFF
Blue LED ON
PCI Express Hot Plug test

- The PCI Express Hot Plug on the MTCA depends on:
  1. Linux Hot Plug Driver
  2. MCH PCI Express Switch with the Hot Plug controller
PCI Express Hot Plug test

- The PCI Express Hot Plug on the MTCA depends on:
  1. Linux Hot Plug Driver
  2. MCH PCI Express Switch with the Hot Plug controller

1. Hot Plug generated by the Hardware (Hardware Hot Plug)
   - Hot Plug triggered by pulling/pushing Module Latch
   - System’s and Device Driver Nodes created/deleted
   - MCH turns AMC Power ON/OFF

2. Hot Plug generated from the user side (Soft Hot Plug)
   - Hot Plug triggered by writing 1/0 to *power* file
   - System’s and Device Driver Nodes created/deleted
   - The AMC module remains powered
PCI Express Hot Plug test

Go to PCIe check

Is device listed in lspci

NO

Enable pciehp

Check pciehp driver

NO

Check MCH

NO

Go to Soft Hotplug check

YES

YES

Check /sys/bus/pci/slots and power file

Check MCH

GO to Hard Hotplug check

Enable device (write 0 to “power” file)
Run lspci

YES

Check kernel log
Call OS expert

NO

Check kernel log
Call MCH expert

GO to Hard Hotplug check

Enable device (write 1 to “power” file)
Run lspci

YES

Check kernel log
Call OS expert

NO

Check kernel log
Call MCH expert

GO to Hard Hotplug check

Push the AMC handle
Run lspci

YES

Check kernel log
Call MCH expert

NO

Use Hotplug

NO

Pull out the AMC handle
Run lspci

YES

Check kernel log
Call MCH expert
PCI Express Hot Plug test

- The PCI Express Hot Plug on the MTCA depends on:
  1. Linux Hot Plug Driver
  2. MCH PCI Express Switch with the Hot Plug controller

- To enable the PCI Express Hot Plug we have to:
  1. Enable Hot Plug Controller of the MCH PCI Express Switch for all ports connected to crate slots
PCI Express Hot Plug test

- The PCI Express Hot Plug on the MTCA depends on:
  1. Linux Hot Plug Driver
  2. MCH PCI Express Switch with the Hot Plug controller

- To enable the PCI Express Hot Plug we have to:
  1. Enable Hot Plug Controller of the MCH PCI Express Switch for all ports connected to crate slots
  2. Enable Linux PCI Express Hot Plug Driver

  - The PCI Express Hot Plug Driver by default is not enabled. To load the driver the following boot parameters have to be setted
    - UBUNTU 10 (Kernel version up to 2.8x)
      - `pciehp.pciehp_force=1 pciehp.pciehp_debug=1`
      - add these parameters in `/boot/grub/menu.lst` file
      - reboot the CPU
    - UBUNTU 12 (Kernel version 3.0 ->)
      - `pciehp.pciehp_force=1 pciehp.pciehp_debug=1 pcie_ports=native`
      - add these parameters in `/etc/default/grub` file and call `update-grub`
      - reboot the CPU
PCI Express Hot Plug test

Checking Hot Plug Driver and PCI Express Switch

- PCI Express Hot Plug driver creates subdirectories in `/sys/bus/pci/slots` for every existing PCI Express slots. In case of MTCA for every physical slot. The name of each directory is the **physical slot number**.
PCI Express Hot Plug test

Checking Hot Plug Driver and PCI Express Switch

- PCI Express Hot Plug driver creates subdirectories in `/sys/bus/pci/slots` for every existing PCI Express slots. In case of MTCA for every physical slot. The name of each directory is the **physical slot number**

Check the `/sys/bus/pci/slots` directory:
- The directors is empty
  - `root@hostname:~# ls /sys/bus/pci/slots`
  - ..
- PCI Express Hot Plug Driver is not loaded
  - Check MCH configuration and OS boot parameters
PCI Express Hot Plug test

Checking Hot Plug Driver and PCI Express Switch

- PCI Express Hot Plug driver creates subdirectories in `/sys/bus/pci/slots` for every existing PCI Express slots. In case of MTCA for every physical slot. The name of each directory is the **physical slot number**

Check the `/sys/bus/pci/slots` directory:
- The directories is empty
  - `root@hostname:~# ls /sys/bus/pci/slots`
    - **PCI Express Hot Plug Drivar is not loaded**
      - Check MCH configuration and OS boot parameters
- Strange numbers (subdirectories names)
  - `root@hostname:~# ls /sys/bus/pci/slots`
    - `0 9 17 (could not be physical slot number 0 and slot number 17 in 12 slots crate)`
      - Wrong PCI Express Switch configuration (**Switch’s Port Number and Slot number are same**)
      - Hot Plug Controller of the PCI Express Switch is not enabled
PCI Express Hot Plug test

Checking Hot Plug Driver and PCI Express Switch

- PCI Express Hot Plug driver creates subdirectories in `/sys/bus/pci/slots` for every existing PCI Express slots. In case of MTCA for every physical slot. The name of each directory is the physical slot number.

Check the `/sys/bus/pci/slots` directory:
- OK
  - `root@hostname:~# ls /sys/bus/pci/slots`
    
    ```
    ... 10 11 12 2 3 4 5 6 7 8 9  
    (CPU in Slot 1)
    ```
PCI Express Hot Plug test

Checking Hot Plug Driver and PCI Express Switch

- PCI Express Hot Plug driver creates subdirectories in `/sys/bus/pci/slots` for every existing PCI Express slots. In case of MTCA for every physical slot. The name of each directory is the **physical slot number**

Check the `/sys/bus/pci/slots` directory:

- OK
  - root@hostname:~# ls /sys/bus/pci/slots
    
    ```
    10 11 12 2 3 4 5 6 7 8 9
    ```
    (CPU in Slot 1)
    
    There is file `power` in each subdirectory. In this file written 1 if there is AMC module in the current slot.

  - root@hostname:~# ls /sys/bus/pci/slots/8
    
    ```
    adapter address attention cur_bus_speed lutch max_bus_speed module power
    ```
  - root@hostname:~# cat /sys/bus/pci/slots/8/power
    
    ```
    1
    ```
PCI Express Hot Plug test

Checking Hot Plug Driver and PCI Express Switch

- PCI Express Hot Plug driver creates subdirectories in `/sys/bus/pci/slots` for every existing PCI Express slots. In case of MTCA for every physical slot. The name of each directory is the **physical slot number**

Check the `/sys/bus/pci/slots` directory:
- OK
  - `root@hostname:~# ls /sys/bus/pci/slots
    .. 10 11 12 2 3 4 5 6 7 8 9  (CPU in Slot 1)`
  - There is file `power` in each subdirectory. In this file written 1 if there is AMC module in the current slot.

- `root@hostname:~# ls /sys/bus/pci/slots/8
  .. adapter address attention cur_bus_speed lutch max_bus_speed module power`
- `root@hostname:~# cat /sys/bus/pci/slots/8/power
  1`
  - If there is AMC module in the Slot but reading from the `power` file returns 0 check MCH configuration
  - Usually Hot Plug Controller of the MCH PCI Express Switch is not enabled
PCI Express Hot Plug test

Checking Soft Hot Plug

Use `/var/log/kern.log` to watch kernel messages and `lspci` to check PCI Express Device

1. Check the module using `lspci`
2. Trigger Hot Plug writing 0 to `power` file
   • `echo 0 > /sys/bus/pci/slots/6/power`
**PCI Express Hot Plug test**

### Checking Soft Hot Plug

Use `/var/log/kern.log` to watch kernel messages and `lspci` to check PCI Express Device

1. Check the module using `lspci`
2. Trigger Hot Plug writing 0 to `power` file
   - `echo 0 > /sys/bus/pci/slots/6/power`
3. Check kernel log file and use `lspci -H1` to check the module is gone

```
kernel: pciehp 0000:04:09.0:pcie24: disable_slot: physical_slot = 6
kernel: pciehp 0000:04:09.0:pcie24: pciehp_unconfigure_device:
domain:bus:dev = 0000:0a:00
kernel: REMOVE CALLED
kernel: REMOVE: UNMAPPING MEMORYs
kernel: PCIEDEV_REMOVE: DESTROY DEVICE MAJOR 247 MINOR 0
```
 PCI Express Hot Plug test

Checking Soft Hot Plug
Use `/var/log/kern.log` to watch kernel messages and `lspci` to check PCI Express Device

1. Check the module using `lspci`
2. Trigger Hot Plug writing 0 to `power` file
   - `echo 0 > /sys/bus/pci/slots/6/power`
3. Check kernel log file and use `lspci -H1` to check the module is gone
4. Enable module writing 1 to the `power` file
   - `echo 1 > /sys/bus/pci/slots/6/power`

kernel: pciehp 0000:04:09.0:pcie24: disable_slot: physical_slot = 6
kernel: pciehp 0000:04:09.0:pcie24: pciehp_unconfigure_device:
domain:bus:dev = 0000:0a:00
kernel: REMOVE CALLED
kernel: REMOVE: UNMAPPING MEMORYs
kernel: PCIEDEV_REMOVE: DESTROY DEVICE MAJOR 247 MINOR 0
PCI Express Hot Plug test

**Checking Soft Hot Plug**

Use `/var/log/kern.log` to watch kernel messages and `lspci` to check PCI Express Device

1. Check the module using `lspci`
2. Trigger Hot Plug writing 0 to `power` file
   - `echo 0 > /sys/bus/pci/slots/6/power`
3. Check kernel log file and use `lspci-H1` to check the module is gone
4. Enable module writing 1 to the `power` file
   - `echo 1 > /sys/bus/pci/slots/6/power`
5. Check kernel log file and use `lspci` to check the module is in

The Software side works.

Checking Hardware part

```
kernel: pciehp 0000:04:09.0:pcie24: disable_slot: physical_slot = 6
kernel: pciehp 0000:04:09.0:pcie24: pciehp_unconfigure_device: domain:bus:dev = 0000:0a:00
kernel: REMOVE CALLED
kernel: REMOVE: UNMAPPING MEMORYs
kernel: PCIEDEV_REMOVE: DESTROY DEVICE MAJOR 247 MINOR 0
```

```
kernl:pciehp 0000:04:09.0:pcie24: __pciehp_link_set: lnk_actl = 0
kernel:pciehp 0000:04:09.0:pcie24: pciehp_green_led_blink: SLOTCTRL 80 write cmd 200
kernel:PCIEDEV_PROBE CALLED
kernel:pciedev 0000:0a:00.0: enabling device (0000 -> 0002)
kernl:PCIEDEV_PROBE: mem_region 0 address C0000000 SIZE 3FFFFFF FLAG 40200
kernel:PCIEDEV: mem_region 1 address C4000000
kernel:PCIEDEV: mem_region 2 address C8000000
kernel:PCIEDEV_PROBE: CREAT DEVICE MAJOR 247 MINOR 0 F_NAME pciedevs6 DEV_NAME
```
PCI Express Hot Plug test

Checking Hardware triggered Hot Plug
Use /var/log/kern.log to watch kernel messages and lspci to check PCI Express Device

1. Check the module using \texttt{lspci –H1}
2. Pull out the AMC handle
PCI Express Hot Plug test

Checking Hardware triggered Hot Plug
Use /var/log/kern.log to watch kernel messages and lspci to check PCI Express Device

1. Check the module using **lspci -H1**
2. Pull out the AMC handle
3. Check kernel log file and use **lspci -H1** to check the module is gone

```markdown
kernel:pciehp 0000:04:09.0:pcie24: pcie_isr: intr_loc 1
kernel:pciehp 0000:04:09.0:pcie24: Attention button interrupt received
kernel:pciehp 0000:04:09.0:pcie24: Button pressed on Slot(6)
kernel:pciehp 0000:04:09.0:pcie24: PCI slot #6 - powering off due to button press.
kernel:PCIEDEV_REMOVE: SLOT 6 DEV 257949696
kernel:REMOVE: UNMAPPING MEMORYs
kernel:PCIEDEV_REMOVE: DESTROY DEVICE MAJOR 246 MINOR 0
```
 PCI Express Hot Plug test

Checking Hardware triggered Hot Plug
Use /var/log/kern.log to watch kernel messages and lspci to check PCI Express Device

1. Check the module using lspci-H1
2. Pull out the AMC handle
3. Check kernel log file and use lspci –H1 to check the module is gone
4. Switch ON the module pushing AMC Handle

kernel:pciehp 0000:04:09.0:pcie24: pcie_isr: intr_loc 1
kernel:pciehp 0000:04:09.0:pcie24: Attention button interrupt received
kernel:pciehp 0000:04:09.0:pcie24: Button pressed on Slot(6)
kerneld:pciehp 0000:04:09.0:pcie24: PCI slot #6 - powering off due to button press.
kerneld:PCIEDEV_REMOVE: SLOT 6 DEV 257949696
kernel:REMOVE: UNMAPPING MEMORYs
kernel:PCIEDEV_REMOVE: DESTROY DEVICE MAJOR 246 MINOR 0
PCI Express Hot Plug test

Checking Hardware triggered Hot Plug

Use `/var/log/kern.log` to watch kernel messages and `lspci` to check PCI Express Device

1. Check the module using `lspci -H1`
2. Pull out the AMC handle
3. Check kernel log file and use `lspci -H1` to check the module is gone
4. Switch ON the module pushing AMC Handle
5. Check kernel log file and use `lspci` to check the module is in

```
kernel:pciehp 0000:04:09.0:pcie24: pcie_isr: intr_loc 1
kernel:pciehp 0000:04:09.0:pcie24: Attention button interrupt received
kernel:pciehp 0000:04:09.0:pcie24: Button pressed on Slot(6)
kernel:pciehp 0000:04:09.0:pcie24: PCI slot #6 - powering off due to button press.
kernel:PCIEDEV_REMOVE: SLOT 6 DEV 257949696
kernel:REMOVE: UNMAPPING MEMORYs
kernel:PCIEDEV_REMOVE: DESTROY DEVICE MAJOR 246 MINOR 0

kernel:pciehp 0000:04:09.0:pcie24: pcie_isr: intr_loc 1
kernel:pciehp 0000:04:09.0:pcie24: Attention button interrupt received
kernel:pciehp 0000:04:09.0:pcie24: Button pressed on Slot(6)
kernel:pciehp 0000:04:09.0:pcie24: PCI slot #6 - powering on due to button press.
kernel:pciehp 0000:04:09.0:pcie24: pciehp_check_link_status:
link_status = 6041
kernel:PCIEDEV_PROBE CALLED
kernel:PCIEDEV_PROBE: mem_region 0 address C0000000 SIZE 3FFFFFF FLAG 40200
kernel:PCIEDEV_PROBE: CREATE DEVICE MAJOR 246 MINOR 0 F_NAME pciedevs6 DEV_NAME
```
PCI Express Hot Plug test

Checking Hardware triggered Hot Plug

Use /var/log/kern.log to watch kernel messages and lspci to check PCI Express Device

1. Check the module using lspci –H1
2. Pull out the AMC handle
3. Check kernel log file and use lspci –H1 to check the module is gone
4. Switch ON the module pushing AMC Handle
5. Check kernel log file and use lspci –H1 to check the module is in
6. Run lspci –H1 with –vvv option to check are the boards memories mapped
7. Check Device Driver file in /dev
8. Try to access to the Device using Device Driver
PCI Express Hot Plug test

Checking Hardware triggered Hot Plug
Use /var/log/kern.log to watch kernel messages and lspci to check PCI Express Device

1. Check the module using lspci -H1
2. Pull out the AMC handle
3. Check kernel log file and use lspci –H1 to check the module is gone
4. Switch ON the module pushing AMC Handle
5. Check kernel log file and use lspci –H1 to check the module is in
6. Run lspci –H1 with –vvv option to check the boards memories mapped
7. Check Device Driver file in /dev
8. Try to access to the Device using Device Driver

According of the kernel log the module is ON but not listed in lspci
• Module initialisation is slow, try to enable it writing 1 to the power file

kernel:pciehp 0000:04:09.0:pcie24: pcie_isr: intr_loc 1
kernel:pciehp 0000:04:09.0:pcie24: Attention button interrupt received
kernel:pciehp 0000:04:09.0:pcie24: Button pressed on Slot(6)
kernel:pciehp 0000:04:09.0:pcie24: PCI slot #6 - powering off due to button press.
kernell:PCIEDEV_REMOVE: SLOT 6 DEV 257949696
kernel:REMOVE: UNMAPPING MEMORYs
kernel:PCIEDEV_REMOVE: DESTROY DEVICE MAJOR 246 MINOR 0

kernel:pciehp 0000:04:09.0:pcie24: pcie_isr: intr_loc 1
kernel:pciehp 0000:04:09.0:pcie24: Attention button interrupt received
kernel:pciehp 0000:04:09.0:pcie24: Button pressed on Slot(6)
kernel:pciehp 0000:04:09.0:pcie24: PCI slot #6 - powering on due to button press.
kernell:PCIEDEV_REMOVE: SLOT 6 DEV 257949696
kernel:REMOVE: UNMAPPING MEMORYs
kernel:PCIEDEV_REMOVE: DESTROY DEVICE MAJOR 246 MINOR 0

Module initialisation is slow, try to enable it writing 1 to the power file
PCI Express Hot Plug test

enjoy PCIe and Hot Plug...

Thank You