PCI Test User Guide

Author: Kishon Vijay Abraham I <kishon@ti.com>

This document is a guide to help users use pci-epf-test function driver and pci_endpoint_test host driver for testing PCI. The list of steps to be followed in the host side and EP side is given below.

Endpoint Device

Endpoint Controller Devices

To find the list of endpoint controller devices in the system:

```
# ls /sys/class/pci_epc/
  51000000.pcie_ep
```

If PCI_ENDPOINT_CONFIGFS is enabled:

```
# ls /sys/kernel/config/pci_ep/controllers
  51000000.pcie_ep
```

Endpoint Function Drivers

To find the list of endpoint function drivers in the system:

```
# ls /sys/bus/pci-epf/drivers
  pci_epf_test
```

If PCI_ENDPOINT_CONFIGFS is enabled:

```
# ls /sys/kernel/config/pci_ep/functions
  pci_epf_test
```

Creating pci-epf-test Device

PCI endpoint function device can be created using the configfs. To create pci-epf-test device, the following commands can be used:

```
# mount -t configfs none /sys/kernel/config
# cd /sys/kernel/config/pci_ep/
# mkdir functions/pci_epf_test/func1
```

The "mkdir func1" above creates the pci-epf-test function device that will be probed by pci_epf_test driver.

The PCI endpoint framework populates the directory with the following configurable fields:

```
# ls functions/pci_epf_test/func1
  baseclass_code        interrupt_pin   progif_code     subsys_id
  cache_line_size       msi_interrupts  revid           subsys_vendorid
  deviceid              msix_interrupts subclass_code   vendorid
```

The PCI endpoint function driver populates these entries with default values when the device is bound to the driver. The pci-epf-test driver populates vendorid with 0xffff and interrupt_pin with 0x0001:

```
# cat functions/pci_epf_test/func1/vendorid
  0xffff
# cat functions/pci_epf_test/func1/interrupt_pin
  0x0001
```

Configuring pci-epf-test Device

The user can configure the pci-epf-test device using configfs entry. In order to change the vendorid and the number of MSI interrupts used by the function device, the following commands can be used:

```
# echo 0x104c > functions/pci_epf_test/func1/vendorid
# echo 0xb500 > functions/pci_epf_test/func1/deviceid
# echo 16 > functions/pci_epf_test/func1/msi_interrupts
# echo 8 > functions/pci_epf_test/func1/msix_interrupts
```

Binding pci-epf-test Device to EP Controller

In order for the endpoint function device to be useful, it has to be bound to a PCI endpoint controller driver. Use the configfs to bind the function device to one of the controller driver present in the system:

```
# ln -s functions/pci_epf_test/func1 controllers/51000000.pcie_ep/
```

Once the above step is completed, the PCI endpoint is ready to establish a link with the host.

Start the Link

In order for the endpoint device to establish a link with the host, the _start_ field should be populated with '1':

```
# echo 1 > controllers/51000000.pcie_ep/start
```

RootComplex Device

lspci Output

Note that the devices listed here correspond to the value populated in 1.4 above:

```
00:00.0 PCI bridge: Texas Instruments Device 8888 (rev 01)
01:00.0 Unassigned class [ff00]: Texas Instruments Device b500
```

Using Endpoint Test function Device

pcitest.sh added in tools/pci/ can be used to run all the default PCI endpoint tests. To compile this tool the following commands should be used:

```
# cd <kernel-dir>
# make -C tools/pci
```

or if you desire to compile and install in your system:

```
# cd <kernel-dir>
# make -C tools/pci install
```

The tool and script will be located in <rootfs>/usr/bin/

pcitest.sh Output

```
# pcitest.sh
BAR tests

BAR0:           OKAY
BAR1:           OKAY
BAR2:           OKAY
BAR3:           OKAY
BAR4:           NOT OKAY
BAR5:           NOT OKAY

Interrupt tests

SET IRQ TYPE TO LEGACY:         OKAY
LEGACY IRQ:     NOT OKAY
SET IRQ TYPE TO MSI:            OKAY
MSI1:           OKAY
MSI2:           OKAY
MSI3:           OKAY
MSI4:           OKAY
MSI5:           OKAY
MSI6:           OKAY
MSI7:           OKAY
MSI8:           OKAY
MSI9:           OKAY
MSI10:          OKAY
MSI11:          OKAY
MSI12:          OKAY
MSI13:          OKAY
MSI14:          OKAY
MSI15:          OKAY
MSI16:          OKAY
MSI17:          NOT OKAY
MSI18:          NOT OKAY
MSI19:          NOT OKAY
MSI20:          NOT OKAY
MSI21:          NOT OKAY
MSI22:          NOT OKAY
MSI23:          NOT OKAY
MSI24:          NOT OKAY
MSI25:          NOT OKAY
MSI26:          NOT OKAY
MSI27:          NOT OKAY
MSI28:          NOT OKAY
MSI29:          NOT OKAY
MSI30:          NOT OKAY
MSI31:          NOT OKAY
MSI32:          NOT OKAY
```

```
SET IRQ TYPE TO MSI-X:          OKAY
MSI-X1:         OKAY
MSI-X2:         OKAY
MSI-X3:         OKAY
MSI-X4:         OKAY
MSI-X5:         OKAY
MSI-X6:         OKAY
MSI-X7:         OKAY
MSI-X8:         OKAY
MSI-X9:         NOT OKAY
MSI-X10:        NOT OKAY
MSI-X11:        NOT OKAY
MSI-X12:        NOT OKAY
MSI-X13:        NOT OKAY
MSI-X14:        NOT OKAY
MSI-X15:        NOT OKAY
MSI-X16:        NOT OKAY
[...]
MSI-X2047:      NOT OKAY
MSI-X2048:      NOT OKAY

Read Tests

SET IRQ TYPE TO MSI:            OKAY
READ (      1 bytes):           OKAY
READ (   1024 bytes):           OKAY
READ (   1025 bytes):           OKAY
READ (1024000 bytes):           OKAY
READ (1024001 bytes):           OKAY

Write Tests

WRITE (      1 bytes):          OKAY
WRITE (   1024 bytes):          OKAY
WRITE (   1025 bytes):          OKAY
WRITE (1024000 bytes):          OKAY
WRITE (1024001 bytes):          OKAY

Copy Tests

COPY (      1 bytes):           OKAY
COPY (   1024 bytes):           OKAY
COPY (   1025 bytes):           OKAY
COPY (1024000 bytes):           OKAY
COPY (1024001 bytes):           OKAY
```