Configuring PCI Endpoint Using CONFIGFS

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

The PCI Endpoint Core exposes configfs entry (pci_ep) to configure the PCI endpoint function and to bind the endpoint function with the endpoint controller. (For introducing other mechanisms to configure the PCI Endpoint Function refer to [1]).

Mounting configfs

The PCI Endpoint Core layer creates pci_ep directory in the mounted configfs directory. configfs can be mounted using the following command:

```
mount -t configfs none /sys/kernel/config
```

Directory Structure

The pci_ep configfs has two directories at its root: controllers and functions. Every EPC device present in the system will have an entry in the *controllers* directory and every EPF driver present in the system will have an entry in the *functions* directory.

```
/sys/kernel/config/pci_ep/
        .. controllers/
        .. functions/
```

Creating EPF Device

Every registered EPF driver will be listed in controllers directory. The entries corresponding to EPF driver will be created by the EPF core.

```
/sys/kernel/config/pci_ep/functions/
        .. <EPF Driver1>/
                ... <EPF Device 11>/
                ... <EPF Device 21>/
                ... <EPF Device 31>/
        .. <EPF Driver2>/
                ... <EPF Device 12>/
                ... <EPF Device 22>/
```

In order to create a <EPF device> of the type probed by <EPF Driver>, the user has to create a directory inside <EPF DriverN>.

Every <EPF device> directory consists of the following entries that can be used to configure the standard configuration header of the endpoint function. (These entries are created by the framework when any new <EPF Device> is created)

```
.. <EPF Driver1>/
        ... <EPF Device 11>/
                ... vendorid
                ... deviceid
                ... revid
                ... progif_code
                ... subclass_code
                ... baseclass_code
                ... cache_line_size
                ... subsys_vendor_id
                ... subsys_id
                ... interrupt_pin
                ... <Symlink EPF Device 31>/
                ... primary/
                        ... <Symlink EPC Device1>/
                ... secondary/
                        ... <Symlink EPC Device2>/
```

If an EPF device has to be associated with 2 EPCs (like in the case of Non-transparent bridge), symlink of endpoint controller connected to primary interface should be added in 'primary' directory and symlink of endpoint controller connected to secondary interface should be added in 'secondary' directory.

The <EPF Device> directory can have a list of symbolic links (<Symlink EPF Device 31>) to other <EPF Device>. These symbolic links should be created by the user to represent the virtual functions that are bound to the physical function. In the above directory structure <EPF Device 11> is a physical function and <EPF Device 31> is a virtual function. An EPF device once it's linked to another EPF device, cannot be linked to a EPC device.

EPC Device

Every registered EPC device will be listed in controllers directory. The entries corresponding to EPC device will be created by the

EPC core.

```
/sys/kernel/config/pci_ep/controllers/
        .. <EPC Device1>/
                ... <Symlink EPF Device11>/
                ... <Symlink EPF Device12>/
                ... start
        .. <EPC Device2>/
                ... <Symlink EPF Device21>/
                ... <Symlink EPF Device22>/
                ... start
```

The <EPC Device> directory will have a list of symbolic links to <EPF Device>. These symbolic links should be created by the user to represent the functions present in the endpoint device. Only <EPF Device> that represents a physical function can be linked to a EPC device.

The <EPC Device> directory will also have a *start* field. Once "1" is written to this field, the endpoint device will be ready to establish the link with the host. This is usually done after all the EPF devices are created and linked with the EPC device.

```
| controllers/
       | <Directory: EPC name>/
               | <Symbolic Link: Function>
               | start
| functions/
       | <Directory: EPF driver>/
               | <Directory: EPF device>/
                       | vendorid
                       | deviceid
                       | revid
                       | progif_code
                       | subclass_code
                       | baseclass_code
                       | cache_line_size
                       | subsys_vendor_id
                       | subsys_id
                       | interrupt_pin
                       | function
```

[1] Documentation/PCI/endpoint/pci-endpoint.rst