NVIDIA JETSON AGX XAVIER PCIE ENDPOINT SOFTWARE FOR L4T

DA-09380-001 | May 1, 2019

Application Note

DOCUMENT CHANGE HISTORY

DA-09380-001

Date	Author	Revision History
May 1, 2019	Stephen Warren, Jonathan Sachs	Initial release.

TABLE OF CONTENTS

1.0 NVIDIA JETSON AGX XAVIER PCIE ENDPOINT SOFTWARE FOR L4T

This document describes PCIe endpoint software for Linux for Tegra (L4T). It describes the software in L4T Release 32.1.

1.1 ARCHITECTURE

L4T contains the following software support for PCIe endpoint mode:

- Linux kernel device driver for the PCIe endpoint controller.

  This driver configures the PCIe controller as an endpoint, and provides an interface for higher level software to select the configuration that is exposed to the PCIe bus.

  Source code for this driver is available at the following paths in the L4T kernel source package:

  ```
  nvidia/drivers/pci/dwc/pcie-tegra.c
  kernel-4.9/drivers/pci/dwc/
  ```

- Trivial example of a Linux kernel PCIe endpoint function driver.

  This driver provides the configuration of the PCIe endpoint, such as BAR count and size, IRQ count, etc. It also implements any runtime functionality of the endpoint.

  This driver is a minimal example, useful for demonstration purposes only. The driver does not interact with the host or with any other part of the endpoint software at run time. It simply exposes some endpoint RAM to the PCIe bus. The customer is expected to implement their own PCIe endpoint function driver according to the needs of their application.

Source code for this driver is available at the following path in the L4T kernel source package:

```
nvidia/drivers/pci/endpoint/functions/pci-epf-nv-test.c
```

- Linux kernel PCIe endpoint subsystem.

  The PCIe endpoint subsystem provides common PCIe endpoint support code. It binds together endpoint controller drivers and endpoint function drivers.

  Source code for this subsystem is available at the following path in the L4T kernel source package:

  ```
  kernel/kernel-4.9/drivers/pci/endpoint/
  ```

1.2 HARDWARE REQUIREMENTS

You need the following hardware to use PCIe endpoint support:

- A Jetson AGX Xavier running L4T, to act as the PCIe endpoint.
- Another computer system, to act as the PCIe root port. This application note assumes that you will use a second Jetson AGX Xavier running L4T. However, any standard x86-64 PC running Linux will act almost identically.
- Cables to connect the two systems. See the NVIDIA application note *NVIDIA Jetson AGX Xavier PCIe Endpoint Design Guidelines* (DA-09357) for details.

1.3 ASSUMPTIONS

All commands described in this application note must be run as root. Some commands use shell I/O redirection, and will not operate correctly if run using `sudo`.

1.4 FLASHING THE PCIE ENDPOINT JETSON AGX XAVIER SYSTEM

A Jetson system must be flashed in a specific way to enable PCIe endpoint mode. Use the following steps to flash the endpoint system:

1. In the extracted L4T release directory, `edit p2972-0000.conf.common`. Set bit 12 of the ODMDATA value, i.e. change it from `0x9190000` to `0x9191000`.
2. Run the following command to re-flash the system:

   ```
   sudo ./flash.sh jetson-xavier mmcblk0p1
   ```

   Note that this completely erases any data previously stored on the Jetson system.
3. Edit `p2972-0000.conf.common` again and restore the `ODMDATA` property's original value. This ensures that any systems flashed in the future operate in PCIe root port mode.

1.5 CONNECTING AND CONFIGURING THE SYSTEMS

1. Connect the systems using the appropriate PCIe cable.
2. Boot the endpoint Jetson system.
3. Run the following commands to configure and enable PCIe endpoint mode:

   ```
   cd /sys/kernel/config/pci_ep/
   mkdir functions/pci_epf_nv_test/func1
   echo 0x10de > functions/pci_epf_nv_test/func1/vendorid
   echo 0x0001 > functions/pci_epf_nv_test/func1/deviceid
   ln -s functions/pci_epf_nv_test/func1 controllers/141a0000.pcie_ep/
   echo 1 > controllers/141a0000.pcie_ep/start
   ```

   For additional details, read the following file in the L4T kernel source package:

   ```
   kernel-4.9/Documentation/PCI/endpoint/pci-endpoint-cfs.txt
   ```
4. Boot the root port system. You must execute this step after step 3 above.

1.6 TESTING PCIE ENDPOINT SUPPORT

The example PCIe endpoint function driver exposes a page of RAM to the root port system. This allows both the endpoint and root port systems access to a shared page of RAM. This section demonstrates a simple method for transferring data between the two systems through the shared RAM.

On both the root port and endpoint systems, run this command to install required utilities:

```
apt install busybox pciutils
```

On the endpoint system, determine the physical address of the RAM that is accessed via the endpoint's BAR:

```
dmesg|grep pci_epf_nv_test
```

This command will print messages similar to the following:

```
[   38.338101] pci_epf_nv_test pci_epf_nv_test.0: BAR0 RAM phys: 0x4307b8000
[   38.338113] pci_epf_nv_test pci_epf_nv_test.0: BAR0 RAM IOVA: 0xffff0000
[   38.338138] pci_epf_nv_test pci_epf_nv_test.0: BAR0 RAM virt: 0xffffff800b3dc000
```

Make a note of the "BAR0 RAM phys" address.

On the endpoint system, you may access the shared RAM using BusyBox. To read the RAM, enter this command:

```
busybox devmem 0x4307b8000
```

To write the RAM, enter this command:

```
busybox devmem 0x4307b8000 32 0xfa950000
```

To allow the PCIe endpoint to respond to PCIe memory accesses on the root port system, enter this command:

```
setpci -s 0005:01:00.0 COMMAND=0x02
```

Note that this command is required only if no Linux kernel device driver binds to the PCIe device and enables the device.

To determine the PCIe address allocated to the endpoint's BAR on the root port system, enter this command:

```
lspci -v
```

This command prints messages similar to the following:

```
0005:01:00.0 RAM memory: NVIDIA Corporation Device 0001
      Flags: fast devsel, IRQ 255
      Memory at 3a300000 (32-bit, non-prefetchable) [disabled] [size=64K]
      Memory at 1c00000000 (64-bit, prefetchable) [disabled] [size=128K]
      Memory at 3a200000 (64-bit, non-prefetchable) [disabled] [size=1M]
      Capabilities: [40] Power Management version 3
      Capabilities: [50] MSI: Enable- Count=1/1 Maskable+ 64bit-
      Capabilities: [70] Express Endpoint, MSI 00
      Capabilities: [b0] MSI-X: Enable- Count=8 Masked-
      Capabilities: [100] Advanced Error Reporting
      Capabilities: [148] #19
      Capabilities: [168] #26
      Capabilities: [190] #27
      Capabilities: [1b8] Latency Tolerance Reporting
      Capabilities: [1c0] L1 PM Substates
      Capabilities: [1d0] Vendor Specific Information: ID=0002 Rev=4 Len=100 <?>
      Capabilities: [2d0] Vendor Specific Information: ID=0001 Rev=1 Len=038 <?>
      Capabilities: [308] #25
      Capabilities: [314] Precision Time Measurement
      Capabilities: [320] Vendor Specific Information: ID=0003 Rev=1 Len=054 <?>
```

Make a note of the first “Memory at” address (the BAR 0 address). The CPU may access this address directly on the root port system. Any such access modifies RAM on the endpoint system. To read the RAM, enter this command:

```
busybox devmem 0x3a300000
```

To write the RAM, enter this command:

```
busybox devmem 0x3a300000 32 0xfa950000
```

To test bidirectional data transfer, enter this sequence of commands:

- On the endpoint system:

  ```
  busybox devmem 0x4307b8000 32 0x98765432
  ```

- On the root port system:

  ```
  busybox devmem 0x3a300000
  ```

Observe that the memory location contains the value `0x98765432`, which was written by the endpoint system.

- Root port system:

```
busybox devmem 0x3a300000 32 0x12345678
```

- Endpoint system:

```
busybox devmem 0x4307b8000
```

Observe that the memory location contains the value `0x12345678`, which was written by the root port system.

www.nvidia.com