Adaptation and Bringup for Jetson AGX Orin

Deployment Guide

DU-10991-001 | June 2022

Table of Contents

Jetson AGX Orin Platform Adaptation and Bring-Up

This guide describes how to port NVIDIA® Jetson™ Linux Driver Package (L4T) From NVIDIA® Jetson AGX Orin™ Developer Kit to another hardware platform.

The examples described include code for the Jetson AGX Orin Developer Kit (P3730).

Refer to T234 BCT (Boot configuration Table) Deployment Guide for information about customizing the configuration files.

Board Configuration

The Jetson AGX Orin Developer Kit consists of a P3701 System on Module (SOM) that is connected to a P3737 carrier board. Part number P3730 designates the complete Jetson AGX Orin Developer Kit. The SOM and carrier board each have an EEPROM where the board ID is saved. The SOM can be used without any software configuration modifications.

The SOM that is sold to be incorporated with customer products has a Thermal Transfer Plate (TTP) that is ready to accept a customer-provided thermal solution. The module that is shipped as part of the Developer Kit does not have TTP, but it has a thermal solution that is designed for the Developer Kit. This thermal solution **must not** be removed from the module.

Before you use the SOM with a carrier board other than P3737, change the kernel device tree, the MB1 configuration, the MB2 Configurations, the ODM data, and the flashing configuration to include configuration for the new carrier board instead of for P3737. The next section provides more information about the changes.

Board Naming

To support a Jetson AGX Orin module with your carrier board, you must assign the module/carrier board combination with a lowercase alphanumeric name. The name can include hyphens (-) and underscores (_) but not spaces. Some examples of valid names are:

```
p3730-0000-devkit
devboard
```

The name you select appears in:

- Filenames and pathnames
- User-visible device tree filenames

This name is also exposed to the user through various Linux kernel `proc` files.

Note:	**In this section, <board> represents your board name.**

You must also select a similarly constructed vendor name, and the same character set rules apply. Here is an example:

`nvidia`

In this section, `<vendor>` represents your vendor name.

Note:	**Do not reuse and modify the existing NVIDIA Jetson AGX Orin Developer Kit code without first selecting and using your own board name. If you do not use your own board name, it will not be obvious to Jetson AGX Orin users whether the modified source code supports the original Jetson AGX Orin Developer Kit board or your board.**

Placeholders in the Porting Instructions

Placeholders are used throughout this topic, and you can substitute an appropriate value for each placeholder when executing commands.

- `<function>` is a functional module name, which might be `power-tree`, `pinmux`, `sdmmc-drv`, `keys`, `comm` (Wi-Fi/Bluetooth®), `camera`, and so on.
- `<board>` is a name you have selected to represent your platform.

  For example, `P3730` is the name of the Jetson AGX Orin Developer Kit. NVIDIA `<board>` names use lower case letters.
- `<version>` is a board version number, such as `a00`.

  Files for NVIDIA reference boards include a version number, and files for customer platforms are not required to include a version number.
- `<vendor>` is the name of your organization or the name of the vendor for your board.
- `<root>` is the device that holds the root file system for the platform.

  The supported value is `emmc`.

Camera Connector Pin Differences

The following table provides information about the camera connector pin differences between the Jetson AGX Orin module and the earlier NVIDIA Jetson AGX Xavier™ modules.

Pin	AGX Xavier	AGX Orin	Notes
1	CSI0	CSI0	Equivalent
2	CSI1	CSI1	Equivalent
3	CSI0	CSI0	Equivalent
4	CSI1	CSI1	Equivalent
5	GND	GND	Equivalent
6	GND	GND	Equivalent
7	CSI0	CSI0	Equivalent
8	CSI1	CSI1	Equivalent
9	CSI0	CSI0	Equivalent
10	CSI1	CSI1	Equivalent
11	GND	GND	Equivalent
12	GND	GND	Equivalent
13	CSI0	CSI0	Equivalent
14	CSI1	CSI1	Equivalent
15	CSI0	CSI0	Equivalent
16	CSI1	CSI1	Equivalent
17	GND	GND	Equivalent
18	GND	GND	Equivalent
19	CSI2	CSI2	Equivalent
20	CSI3	CSI3	Equivalent
21	CSI2	CSI2	Equivalent
22	CSI3	CSI3	Equivalent
23	GND	GND	Equivalent
24	GND	GND	Equivalent
25	CSI2	CSI2	Equivalent
26	CSI3	CSI3	Equivalent
27	CSI2	CSI2	Equivalent
28	CSI3	CSI3	Equivalent
29	GND	GND	Equivalent
30	GND	GND	Equivalent
31	CSI2	CSI2	Equivalent
32	CSI3	CSI3	Equivalent

Pin	AGX Xavier	AGX Orin	Notes
33	CSI2	CSI2	Equivalent
34	CSI3	CSI3	Equivalent
35	GND	GND	Equivalent
36	GND	GND	Equivalent
37	CSI4	CSI4	Equivalent
38	CSI6	CSI6	Equivalent
39	CSI4	CSI4	Equivalent
40	CSI6	CSI6	Equivalent
41	GND	GND	Equivalent
42	GND	GND	Equivalent
43	CSI4	CSI4	Equivalent
44	CSI6	CSI6	Equivalent
45	CSI4	CSI4	Equivalent
46	CSI6	CSI6	Equivalent
47	GND	GND	Equivalent
48	GND	GND	Equivalent
49	CSI4	CSI4	Equivalent
50	CSI6	CSI6	Equivalent
51	CSI4	CSI4	Equivalent
52	CSI6	CSI6	Equivalent
53	GND	GND	Equivalent
54	GND	GND	Equivalent
55	RSVD	RSVD	Equivalent
56	RSVD	RSVD	Equivalent
57	RSVD	RSVD	Equivalent
58	RSVD	RSVD	Equivalent
59	CSI5	CSI5	Equivalent
60	CSI7	CSI7	Equivalent
61	CSI5	CSI5	Equivalent
62	CSI7	CSI7	Equivalent
63	GND	GND	Equivalent
64	GND	GND	Equivalent
65	CSI5	CSI5	Equivalent
66	CSI7	CSI7	Equivalent
67	CSI5	CSI5	Equivalent
68	CSI7	CSI7	Equivalent

Pin	AGX Xavier	AGX Orin	Notes
69	GND	GND	Equivalent
70	GND	GND	Equivalent
71	CSI5	CSI5	Equivalent
72	CSI7	CSI7	Equivalent
73	CSI5	CSI5	Equivalent
74	CSI7	CSI7	Equivalent
75	I2C_GP3 (CAM_I2C)	I2C_GP3 (CAM_I2C)	Equivalent
76	NC	CAM_ERROR1	Camera error line
77	I2C_GP3 (CAM_I2C)	I2C_GP3 (CAM_I2C)	Equivalent
78	NC	CAM_ERROR2	Camera error line
79	GND	GND	Equivalent
80	GND	GND	Equivalent
81	2.8V (AVDD_CAM)	2.8V (AVDD_CAM)	Equivalent
82	2.8V (AVDD_CAM)	2.8V (AVDD_CAM)	Equivalent
83	2.8V (AVDD_CAM)	2.8V (AVDD_CAM)	Equivalent
84	NC	CAM_ERROR3	Camera error line
85	NC	CAM_FRSYNC1	Camera Frame Sync
86	NC	CAM_ERROR4	Camera Frame Sync
87	I2C_GP2	I2C_GP2	Equivalent
88	CAM1_MCLK03	CAM1_MCLK03	Equivalent
89	I2C_GP2	I2C_GP2	Equivalent
90	GPIO15_CAM1_PWDN	GPIO15_CAM1_PWDN	Equivalent
91	CAM0_MCLK02	CAM0_MCLK02	Equivalent
92	GPIO16_CAM1_RST	GPIO16_CAM1_RST	Equivalent
93	CAM0_PWDN	CAM0_PWDN	Equivalent
94	CAM2_MCLK04	CAM2_MCLK04	Equivalent
95	CAM0_RST	CAM0_RST	Equivalent
96	NC	CAM_FRSYNC4	Camera Frame Sync
97	NC	CAM_FRSYNC3	Camera Frame Sync
98	NC	CAM_FRSYNC2	Camera Frame Sync
99	GND	GND	Equivalent
100	GND	GND	Equivalent
101	RSVD	CAM_TE_RSV	

Pin	AGX Xavier	AGX Orin	Notes
102	1.8V (VDD_1V8)	1.8V (VDD_1V8)	Equivalent
103	NC	CAM_INT3	Camera Interrupt
104	NC	CAM_INT4	Camera Interrupt
105	I2C_GP4	I2C_GP9	
106	NC	CAM_INT2	Camera Interrupt
107	I2C_GP4	I2C_GP9	
108	3.3V (VDD_3V3)	3.3V (VDD_3V3)	Equivalent
109	NC	CAM_BACKLIGHT_PWM	Backlight PWM signal This is a reserved function which requires rework to enable. This PWM has been routed to the 40 Pin connector by default
110	3.3V (VDD_3V3)	3.3V (VDD_3V3)	Equivalent
111	NC	CAM_SPI	CAM_SPI is a reserved function which requires rework to enable. This SPI has been routed to 40Pin connector
112	NC	CAM_SPI	
113	NC	CAM_SPI	
114	NC	CAM_SPI	
115	GND	GND	Equivalent
116	GND	GND	Equivalent
117	NC	CAM_INT1	Camera Interrupt
118	3.3V (VDD_3V3)	3.3V (VDD_3V3)	Both Xavier and Orin define this as 3.3V. The older platform such as T186 uses 5V.
119	GPIO25_VDD_SYS_EN	GPIO12_VDD_SYS_EN	Xavier and Orin have different pin mapping to the CVM
120	3.3V (VDD_3V3)	3.3V (VDD_3V3)	Both Xavier and Orin define it as 3.3V. The older platform such as T186 uses 5V.

Root Filesystem Configuration

L4T can use any standard or customized Linux root filesystem (`rootfs`) that is appropriate for their targeted embedded applications.

However, certain settings must be configured in the `rootfs`'s boot-up framework to set default configuration after the boot or some of the core functionalities will not run as expected.

For example:

- The `nv.sh` and `nvfb.sh` boot-up scripts do some platform-specific configuration in the kernel.
- The Xorg and X libraries must be correctly configured for the target device.
- The nvpmodel clock and frequency must be configured for the target device.

These `rootfs` configurations and customizations are provided in this driver package in the `Linux_for_Tegra/nv_tegra/` directory and its subdirectories:

You must incorporate the relevant customization for your target rootfs from this location.

Note: **For the sample Ubuntu root filesystem provided by NVIDIA, this customization is applied using the `Linux_for_Tegra/apply_binaries.sh` script.**

MB1 Configuration Changes

From version t234 onwards, multiple `.dts/dtsi` files define boot time configuration of the hardware, these files are applied by Bootloader. The MB1 boot configuration tables are available at:

```
<l4t_top>/bootloader/t186ref/BCT
```

For more details about the BCT, refer to the Jetson Developer Guide. Click **Software Feature in-depth → Bootloader → T23x boot configuration table** for information about configuring BCT.

Pinmux Changes

If your board schematic differs from the schematic for the Jetson AGX Orin Developer Kit board, you must change the pinmux configuration applied by the software.

To define your board's pinmux configuration, download the Jetson AGX Orin pinmux table from the Jetson Download Center. Ensure that you download the correct version of the table for your SOM. The table is a spreadsheet and:

- Shows the locations and default pinmux settings.
- Defines the pinmux settings in the source code or device tree

To access the spreadsheet, go to

https://developer.nvidia.com/embedded/downloads.

You must customize the spreadsheet for the configuration of your board and click **Generate DT File** to generate the following files:

- `pinmux.dtsi`
- `gpio.dtsi`
- `padvoltage.dtsi`

You need to add these files to `<l4t_top>/bootloader/t186ref/BCT/` and ensure that you point to the new files in the new `board.conf` file that you created for your board. Refer to Flashing the Build Image for more information.

Identifying the GPIO Number

If you designed your own carrier board, to translate from SOM connector pins to actual GPIO numbers, you must understand the following GPIO mapping formula. The translated GPIO numbers can be controlled by the driver.

Because the Jetson module dynamically registers GPIOs, search the kernel messages to check the GPIO allocation ranges for each GPIO group. The command and resulting output are similar to the following:

```
root@jetson:/home/ubuntu# dmesg | grep gpiochip
[    5.726492] gpiochip0: registered GPIOs 348 to 511 on tegra234-gpio
[    5.732478] gpiochip1: registered GPIOs 316 to 347 on tegra234-gpio-aon
root@jetson:/home/ubuntu#
```

As shown in the output above, there are two Jetson GPIO ports with different base indices:

- **`tegra234-gpio`**, at base index 348
- **`tegra234-gpio-aon`**, at base index 316

You can check the GPIO number in one of the following ways:

- Using a calculation.
  a. B**efore you get started**, you need to know how you plan to configure the offset at each available port.

     Here is the list of the tegra234 GPIO ports and offset mapping:

Port	No of pins	Port Offset
PORT_A	8	0
PORT_B	1	8
PORT_C	8	9
PORT_D	4	17
PORT_E	8	21
PORT_F	6	29
PORT_G	8	35
PORT_H	8	43
PORT_I	7	51
PORT_J	6	58

PORT_K	8	64
PORT_L	4	72
PORT_M	8	76
PORT_N	8	84
PORT_P	8	92
PORT_Q	8	100
PORT_R	6	108
PORT_X	8	114
PORT_Y	8	122
PORT_Z	8	130
PORT_AC	8	138
PORT_AD	4	146
PORT_AE	2	150
PORT_AF	4	152
PORT_AG	8	156

b. Search for the pin details from the Orin pinmux table (see Pinmux Changes).

   For example `SOC_GPIO08`, which is **GPIO3_PB.00**.

c. Identify the port as **B** and the Pin_offset as **0**.
d. Calculate the pin number with the following formula:

   `base + port_offset + pin_offset`.

e. Verify the following values:
   - The base is 348.

     This value comes from the kernel boot log, it is already noted **tegra234-gpio, at base index 348**.

   - `port_offset` of port B = 8

     (This value comes from the tegra234 GPIO port and the offset mapping above.

   - pin_offset = 0
   - Pin number = 348 + 8 + 0 = 356

- Using Kernel debugfs
  a. Search for the gpio pin in the Jetson AGX Orin pinmux table (see Pinmux Changes).

     For example, `SOC_GPIO08`, which is **GPIO3_PB.00.**

b. Follow the gpio debugfs look up that use the port and offset.

```
cat /sys/kernel/debug/gpio | grep PB.00
```

c. The gpio number is mentioned in the first col as **gpio-356.**

```
root@jetson:/home/ubuntu# cat /sys/kernel/debug/gpio | grep PB.00
 gpio-356 (PB.00               )
```

MB2 Configuration Changes

Modifying for EEPROM

EEPROM is an optional component for a customized carrier board. In the case the carrier board is designed without an EEPROM, below modifications will be needed on the MB2 BCT file: Linux_for_Tegra/bootloader/tegra234-mb2-bct-common.dtsi

- cvb_eeprom_read_size = <0x100>

+ cvb_eeprom_read_size = <0x0>

Changing the Pinmux

With the JP 5.0-DP release, you can change the pinmux in the following ways:

- Update the MB1 pinmux BCT.
- To make pinmux changes dynamically, the “devmem” tool can be used to update the pinmux register.

Currently users are not able to use the pinctrl APIs to update pinmux. It is due to the upstreamed GPIO driver not yet supporting calling the pinctrl APIs to make changes in the pinmux registers.

To dynamically make pinmux changes:

1. Get the pinmux register address.
   a. Find the below chapter from the TRM: **System Components → Multi-Purpose I/O Pins and Pin Multiplexing (PinMux) → Pinmux Registers**.
   b. Search for the pin name (for example, `SOC_GPIO37`).
   c. Write down the complete pin name (for example: `PADCTL_G3_SOC_GPIO37_0`) and the Offset (for example, `SOC_GPIO37: Offset = 0x80`).
   d. Go to the Jetson Orin Technical Reference Manual, and in *Table 1-15: Pad Control Grouping* table, find the `G3 pad control block = PADCTL_A0` entry.
   e. On the Memory Architecture page, click **Memory Mapped I/O → Address Map**.
   f. Search for `PADCTL_A0`.

- The base address is `PADCTL_A0 = 0x02430000`, and the pinmux register address is *PADCTL base address + offset*.
- For example, `SOC_GPIO37 Pinmux register address = 0x02430000 + 0x80 = 0x02430080`.

2. Get current register value in device using devmem tool:
   a. Install `devmem`.

   ```
   $ sudo apt-get install busybox
   ```

   ```
   $ busybox devmem <32-bit address>
   ```

   For example, `devmem 0x02430080`

3. To use pin as GPIO, update following fields in the PADCTL register:

   Find register information from the Pinmux Registers section in TRM as mentioned above.

   a. Set GPIO: `Bit 10 = 0.`
   b. For the output, set `Bit 4 = 0 ; Bit 6 = 0.`
   c. For Input, set `Bit 4 = 1 ; Bit 6 = 1.`
4. Use the `devmem` tool to set the register, for example, `busybox devmem 0x02430080 w 0x050`.
5. Check and ensure that the register value is set accordingly.

A Complete Example

The following example shows you how to update `Pin MCLK05: SOC_GPIO33: PQ6`:

1. The Pinmux register base address = 0x2430000
   - Offset = 0x70
   - Pinmux register address = 0x2430070
2. Run the following command.

   ```
   $ busybox devmem 0x02430070
   ```

   The output value is `0x00000054`.

3. To set the pin as the output, run the following command.

   ```
   $ busybox devmem 0x02430070 w 0x004
   ```

4. Read back to confirm:

   ```
   $ busybox devmem 0x02430070
   ```

The following commands are used to set the direction of the GPIO controller between the input and the output:

Identify the GPIO number by completing the steps in Identifying the GPIO Number.

```
$ echo 454 > /sys/class/gpio/export
```

```
$ cd /sys/class/gpio/PQ.06

$ echo out > direction

$ cat value

# change value and check
```

Porting the Linux Kernel

We assume that you are using a P3701 SOM that is connected to a P3737 carrier board, and which has not been modified. The eMMC, PMIC, and DDR are the same with the same routing of lines. The modifications that you make are for the carrier board. Consequently, based on the peripherals on your carrier board, you can modify the `.dts` files by disabling/enabling the controllers and changing the supplies.

To port the kernel configuration code (the device tree) to your platform, modify one of the distributed configuration files to describe the design of your platform:

The configuration files are available in the following directories:

```
<top>/hardware/nvidia/platform/t23x/
<top>/hardware/nvidia/soc/t23x
```

Here is the final DTB file that is used:

```
tegra234-p3701-0000-p3737-0000.dtb
```

By reading the final DTB file, you see which other `.dtsi` files are referenced by the include statements. Here is a list of the common `.dtsi` files that might be modified to reflect hardware design changes:

Types of Changes	DTSI Filename or Location
Power supply changes	`tegra234-p3701-power-tree.dtsi` `tegra234-power-tree-p3701-0000-p3737-0000.dtsi`
Display panel and node changes	Refer to the Display Configuration and Bring-up section of the topic Kernel Customization.
ODM data based feature configuration	respective overlay dt, for example: • `ufs: tegra234-p3737-overlay-ufs.dts` • `pcie: tegra234-p3737-overlay-pcie.dts`
NVIDIA SoC controller state to enable/disable a controller	soc/t23x/kernel-dts/tegra234-soc/

Verify that no other `.dts` or `.dtsi` file, including these `.dts` files, overrides the changes you make.

As a best practice, create your own set of `.dts` files based on the existing 3737 files and rename your newly created files to the name of your board.

Note: **Use `fdtdump` or `dtc` to generate a .dts from the final .dtb file and check if your changes have taken effect.**

The command usage is:

```
dtc -I dtb -O dts tegra234-p3701-0000-p3737-0000.dtb >
tegra234-p3701-0000-p3737-0000.dtb
```

```
fdtdump dts tegra234-p3701-0000-p3737-0000.dtb >
tegra234-p3701-0000-p3737-0000.dtb
```

PCIe Controller Configuration

The PCIe host controller is based on the Synopsys Designware PCIe intellectual property, so the host inherits the common properties that are defined in the information file at:

```
$(KERNEL_TOP)/Documentation/devicetree/bindings/pci/nvidia,tegra194-pcie.txt
```

PCIe Controller Features

Jetson AGX Orin has the following PCIe controllers with these specifications:

- **Speed**: All controllers support up to Gen4 speed.
- Lane width:
  - C5, C7: up to x8
  - C0, C4, C6, C10: up to x4
  - C8, C9: up to x2
  - C1, C2, C3: x1
- **Controllers**: Controllers C5, C6, C7 and C10 support dual mode, that is, can be configured as endpoints.
- **ASPM**: All controllers support ASPM.

The Jetson AGX Orin devkit default PCIe configuration is:

- C5: x8
- C4: x4
- C1: x1

These PCIe slots available in the Jetson AGX Orin devkit:

- **M.2 Key M**: C4 controller operates in x4. Any M.2 Key M form factor NVMe cards can be connected.
- **M.2 Key E**: C3 controller operates in x1 mode. Any M.2 Key E form factor cards like Wi-Fi can be connected.
- **PCIe slot**: C5 controller operates in x8 mode. Any PCIe card can be connected. The PCIe slot is of x16 size to connect x16 card, but operates in x8 mode.

To enable PCIe in customer CVB design:

1. Select the appropriate UPHY configuration from UPHY Lane Configuration, which suits the CVB design and update ODMDATA accordingly.
2. Enable the appropriate PCIe node from the table below.
3. Add the `pipe2uphy` phandle entries as a `phy` property in the PCIe controller DT node.

   `pipe2uphy` DT nodes are defined in SoC DT at `$(TOP)/hardware/nvidia/soc/t234/kernel-dts/tegra234-soc/tegra234-soc-pcie.dtsi`. Each `pipe2uphy` node is a 1:1 map to the UPHY lanes that are defined in UPHY Lane Configuration.

For information about Jetson AGX Orin specific PCIe controller configuration, see the device tree documentation file at:

```
$(KERNEL_TOP)/Documentation/devicetree/bindings/pci/nvidia,tegra194-pcie.txt
```

This file covers topics that include configuring maximum link speed, link width, and the advertisement of different ASPM states.

Here are the PCIe controller DT nodes:

PCIe controller and mode	PCIe controller DT node
PCIe C0 RP	pcie@14180000
PCIe C1 RP	pcie@14100000
PCIe C2 RP	pcie@14120000
PCIe C3 RP	pcie@14140000
PCIe C4 RP	pcie@14160000
PCIe C5 RP	pcie@141a0000
PCIe C6 RP	pcie@141c0000
PCIe C7 RP	pcie@141e0000
PCIe C8 RP	pcie@140a0000
PCIe C9 RP	pcie@140c0000
PCIe C10 RP	pcie@140e0000
PCIe C5 EP	pcie_ep@141a0000
PCIe C6 EP	pcie_ep@141c0000
PCIe C7 EP	pcie_ep@141e0000
PCIe C10 EP	pcie_ep@140e0000

Example change: PCIe x1 (C0) and PCIe x8 (C7)

```
diff --git
a/hardware/nvidia/platform/t23x/concord/kernel-dts/cvb/tegra234-ethernet-3737-0
000.dtsi
b/hardware/nvidia/platform/t23x/concord/kernel-dts/cvb/tegra234-ethernet-3737-0
000.dtsi

index 05c2ba700..6115dea53 100644

---
a/hardware/nvidia/platform/t23x/concord/kernel-dts/cvb/tegra234-ethernet-3737-0
000.dtsi

+++
b/hardware/nvidia/platform/t23x/concord/kernel-dts/cvb/tegra234-ethernet-3737-0
000.dtsi
```

```
@@ -19,7 +19,7 @@

 / {

       /* MGBE - A */

       ethernet@6810000 {

-            status = "okay";

+            status = "disable";

             nvidia,mac-addr-idx = <0>;

             nvidia,max-platform-mtu = <16383>;

             /* 0=enable, 1=disable */

diff --git
a/hardware/nvidia/platform/t23x/concord/kernel-dts/cvb/tegra234-p3737-pci
e.dtsi
b/hardware/nvidia/platform/t23x/concord/kernel-dts/cvb/tegra234-p3737-pci
e.dtsi

index bc065d35f..8f9f9b617 100644

---
a/hardware/nvidia/platform/t23x/concord/kernel-dts/cvb/tegra234-p3737-pci
e.dtsi

+++
b/hardware/nvidia/platform/t23x/concord/kernel-dts/cvb/tegra234-p3737-pci
e.dtsi



+     pcie@14180000 {

+            status = "okay";

+            phys = <&p2u_hsio_0>;

+            phy-names = "p2u-0";

+     };

+

+     pcie@141e0000 {

+            status = "okay";

+            num-lanes = <8>;

+            phys = <&p2u_gbe_0>, <&p2u_gbe_1>, <&p2u_gbe_2>,
```

```
+                       <&p2u_gbe_3>, <&p2u_gbe_4>, <&p2u_gbe_5>,
+                       <&p2u_gbe_6>, <&p2u_gbe_7>;
+              phy-names = "p2u-0", "p2u-1", "p2u-2", "p2u-3",
+                          "p2u-4", "p2u-5", "p2u-6", "p2u-7";
+      };
```

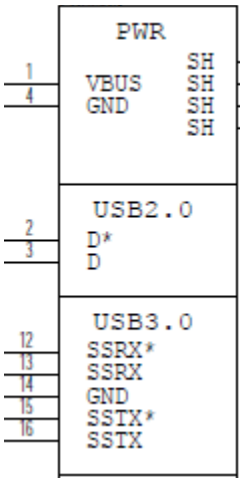
Porting the Universal Serial Bus

The Jetson AGX Orin series can support up to four enhanced SuperSpeed Universal Serial Bus (USB) ports. In some implementations, not all of these ports can be used because of UPHY lane sharing among PCIE, UFS, and XUSB. The Jetson P3737 carrier board is designed and verified for three USB ports. If you designed your own carrier board, verify the UPHY lane mapping and compatibility between P3737 and your custom board by consulting the NVIDIA team.

USB Structure

An enhanced SuperSpeed USB port has nine pins:

- VBUS
- GND
- D+
- D−
- Two differential signal pairs for SuperSpeed data transfer
- One ground (GND_DRAIN) for drain wire termination and managing EMI, RFI, and signal integrity



The D+/D− signal pins connect to UTMI pads. The SSTX/SSRX signal pins connect to UPHY and are handled by a single UPHY lane. As UPHY lanes are shared between PCIE, UFS, and XUSB, UPHY lanes must be assigned according to the custom carrier board's requirements.

Universal Physical Layer Lane Assignment

Universal physical layer **(**UPHY) is a physical I/O interface layer that can serve multiple types of interfaces, for example, USB and PCIe. A UPHY lane can support multiple interface types.

The Jetson P3737 carrier board is designed and verified for three USB3.2 ports. The verified use cases and their UPHY lane assignments are shown in the following table. Note that no mixing or matching of interfaces between the configurations is supported.

Jetson AGX Orin Pin Names	Orin UPHY Block and Lane	Jetson AGX Orin Functions	
		Configuration #1	Configuration #2
UPHY_RX0/TX0	UPHY0, Lane 0	USB 3.2 (P0)	PCIe x1 (C0), RP
UPHY_RX1/TX1	UPHY0, Lane 1	USB 3.2 (P1)	USB 3.2 (P1)
UPHY_RX20/TX20	UPHY0, Lane 2	USB 3.2 (P2)	USB 3.2 (P2)
UPHY_RX21/TX21	UPHY0, Lane 3	PCIe x1 (C1), RP	PCIe x1 (C1), RP
UPHY_RX22/TX22	UPHY0, Lane 4		
UPHY_RX23/TX23	UPHY0, Lane 5		
UPHY_RX10/TX10	UPHY0, Lane 6		
UPHY_RX11/TX11	UPHY0, Lane 7	PCIe x4 (C4), RP	PCIe x4 (C4), RP

Before you design your custom board, refer to the NVIDIA Jetson AGX Orin Series SOC Technical Reference Manual (TRM), the *NVIDIA Jetson AGX Orin Series Product Design Guide* (DG), and contact NVIDIA.

Required Device Tree Changes

This section provides guidance to check schematics and configure USB ports in the device tree. All the examples are based on the design of the Jetson AGX Orin P3737 carrier board.

For a Host-Only Port

This section takes J39, a USB 3.2 type-C connector as an example of a host-only port.

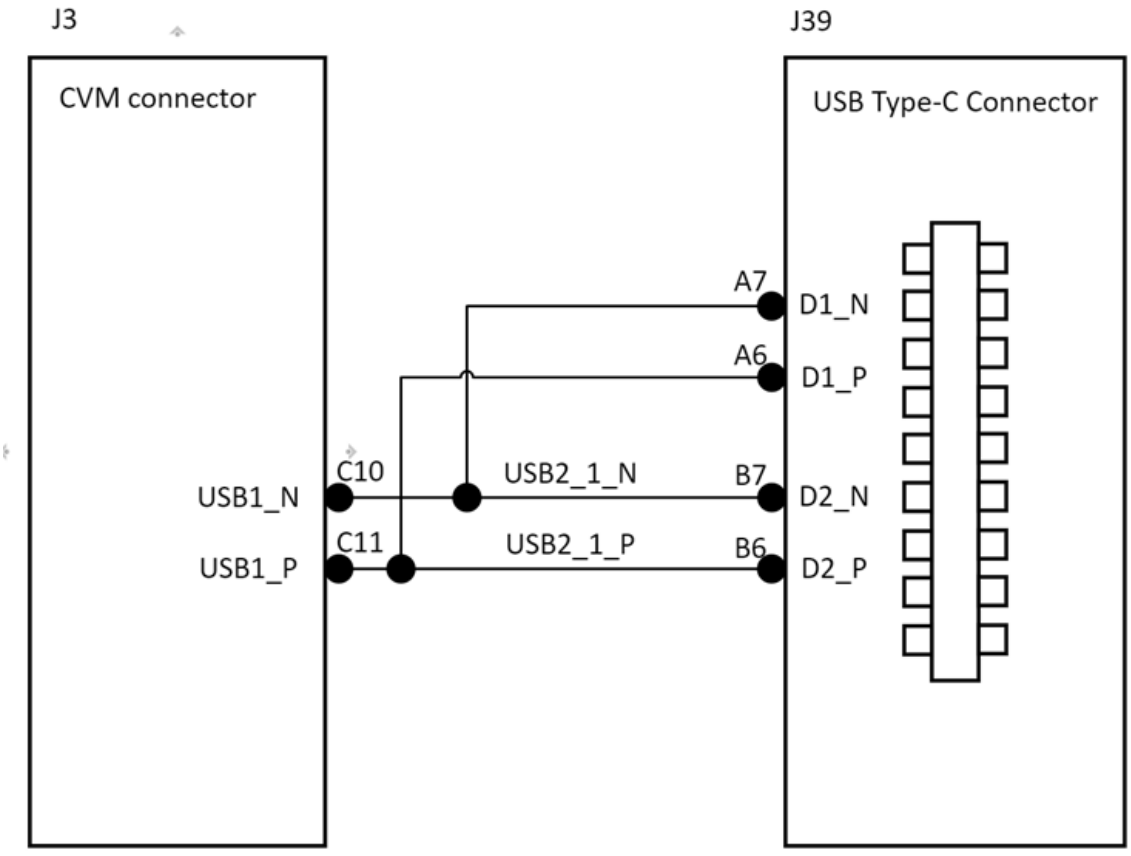
Review the Schematics

Note: **The P3737 carrier board's schematic file, `P3737_A04_Concept_schematics.pdf`, is included in *Jetson AGX Orin Series Developer Kit Carrier Board Design Files (A04)*, available at:**

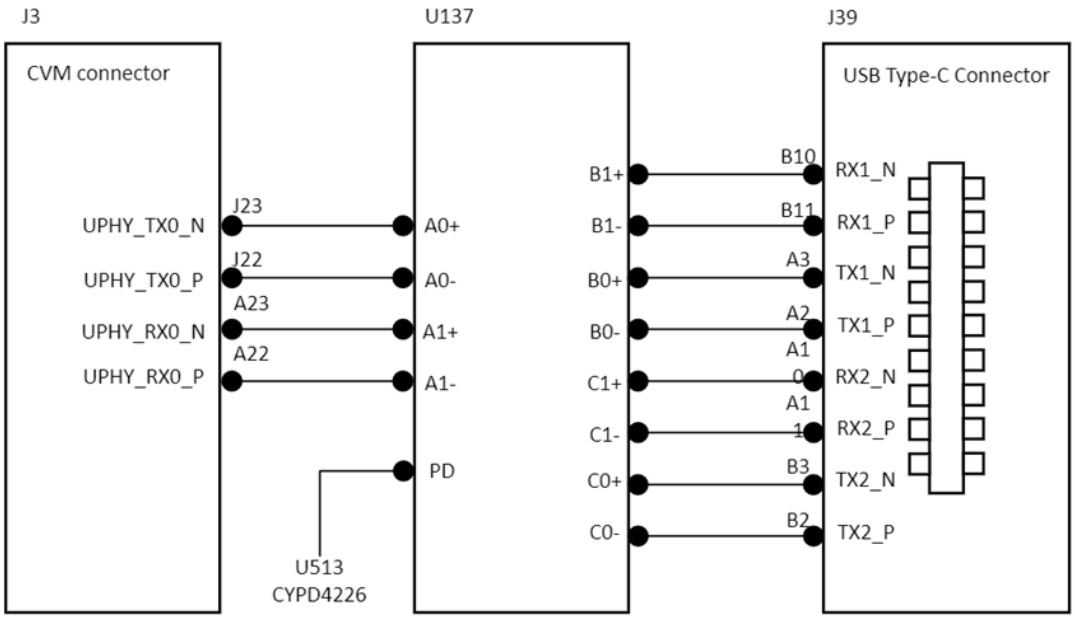
https://developer.nvidia.com/jetson-agx-orin-developer-kit-carrier-board-design-files-a04

- Check the USB connectors on the P3737 carrier board and find the socket location wired to the P3701 SOM.

  USB2.0 signal pins D+/D- (USB2_1_*) wire out from J39 and lead to C10 (USB1_N) and C11 (USB1_P) on the SOM socket.



- USB3.2 differential signal pairs (TX* and RX*) wire out from J39 and lead to J23 (UPHY_TX0_N), J22 (UPHY_TX0_P), A23 (UPHY_RX0_N), and A22 (UPHY_RX0_P) on the SOM socket through U137 and U513, the USB type-C alt mode switch.



Through the schematic, we can conclude that for J39:

- The USB2.0 signal pair is wired to UTMI pad 1 (USB2 port 1).
- The USB3.2 signal pairs are wired to UPHY lane 0 (USB3.2 port 0 according to UPHY lane mapping).

The xusb_padctl Node

The device tree's `xusb_padctl` node follows the conventions of the `pinctrl-bindings.txt` kernel document. It contains two sets of groups named `pads` and `ports`, which describe USB2 and USB3 signals along with parameters and port numbers. The name of each parameter description subnode in `pads` and `ports` must be in the form `<type>-<port_number>`, where `<type>` is "`usb2`" or "`usb3`" and `<port_number>` is the associated port number.

The pads Subnode

- `nvidia,function`: A string containing the name of the function to mux to the pin or group. Must be "xusb".

The ports Subnode

- `mode`: A string that describes USB port capability. A port for USB2 must have this property. It must be one of these values:
  - `host`
  - `device`

    - `OTG`

- `nvidia,usb2-companion`: USB2 port (0/1/2/3) to which the port is mapped. A port for USB3 must have this property.
- `nvidia,oc-pin`: The overcurrent VBUS pin the port is using. The value must be positive or zero.

  Note: **Overcurrent detection and handling for J39 and J40 on the P3737 carrier board are controlled by U513, a Cypress Type-C controller. Therefore, you need not set this property for J39 and J40 USB ports.**

- `vbus-supply`: VBUS regulator for the corresponding UTMI pad. Set to "`&battery_reg`" for a dummy regulator.

  Note: **The VBUS regulators for J39 and J40 are controlled by U513, a Cypress Type-C controller. Therefore, you must set dummy regulators for those ports on the P3737 carrier board.**

- `nvidia,usb3-gen1-only`: A number (1/0) which describes whether or not to limit the port speed to USB3.1 gen1.

  For the detailed information about `xusb_padctl`, refer to the documentation at:

  `https://nv-tegra.nvidia.com/r/plugins/gitiles/linux-5.10/+/refs/heads/l4t/l4t-r34.dp-5.10/Documentation/devicetree/bindings/phy/`

Take J39 (USB3.2 type-C connector) for example and create a pad/port node and property list for J39 based on the device tree structure described above:

```
xusb_padctl: xusb_padctl@3520000 {
  ...
   pads {
         usb2 {
               lanes {
                     ...
                     usb2-1 {
                           nvidia,function = "xusb";
                           status = "okay";
                     };
                     ...
               };
         };
         usb3 {
               lanes {
                     ...
                     usb3-0 {
                           nvidia,function = "xusb";
                           status = "okay";
                     };
                     ...
               };
         };
   };
   ports {
         ...
         usb2-1 {
               mode = "host";
               vbus-supply = <&battery_reg>;
               status = "okay";
         };
         ...
         usb3-0 {
               nvidia,usb2-companion = <1>;
               status = "okay";
         };
         ...
   };
};
```

Under the xHCI Node

The Jetson AGX Orin xHCI controller complies to xHCI specifications, which support both USB 2.0 HighSpeed/FullSpeed/LowSpeed and USB 3.2 SuperSpeed protocols.

- `phys`: Must contain an entry for each entry in `phy-names`.
- `phy-names`: Must include an entry for each PHY used by the controller. Names must be of the form `<type>-<port_number>`, where `<type>` is `"usb2"` or `"usb3"`.

- `nvidia,xusb-padctl`: A pointer to the `xusb-padctl` node.

Refer to the following documentation for more information:

`kernel/kernel-5.10/Documentation/devicetree/bindings/usb/nvidia,tegra-xhci.txt`

Take the J39 USB3.2 type-C connector for example and create an xHCI node and property list for J39 based on the device tree structure described above:

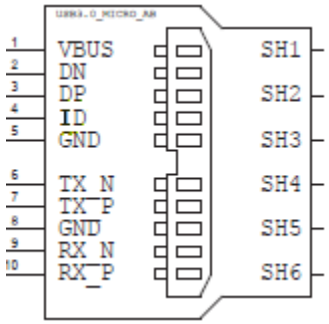
```
tegra_xhci: xhci@3610000 {
    ...
    phys = <&{/xusb_padctl@3520000/pads/usb2/lanes/usb2-1}>,
           <&{/xusb_padctl@3520000/pads/usb3/lanes/usb3-0}>;
    phy-names = "usb2-1", "usb3-0";
    nvidia,xusb-padctl = <&xusb_padctl>;
    status = "okay";
    ...
};
```

For an OTG (On-The-GO) Port

USB On-The-Go, often abbreviated **USB OTG** or just **OTG**, is a specification that allows USB to act as a host or a device in the same port. A USB OTG port can switch back and forth between the roles of host and device.

This section takes J40, USB3.2 type-C connector, as an example of an OTG port.

An OTG port adds a fifth pin to the standard USB connector, called the **ID pin**. An OTG cable has an A-plug on one end and a B-plug on the other end. The A-plug's ID pin is grounded, while the B-plug's ID pin is floating. A device with an A-plug inserted becomes an OTG A-device (host), and a device with a B-plug inserted becomes a B-device (device).



Note: **The roles of J40, the port switch, between the host driver (xHCI) and device driver (xUDC) are controlled by a U513 Cypress Type-C controller and ucsi_ccg driver in the Jetson AGX Orin Developer Kit.**

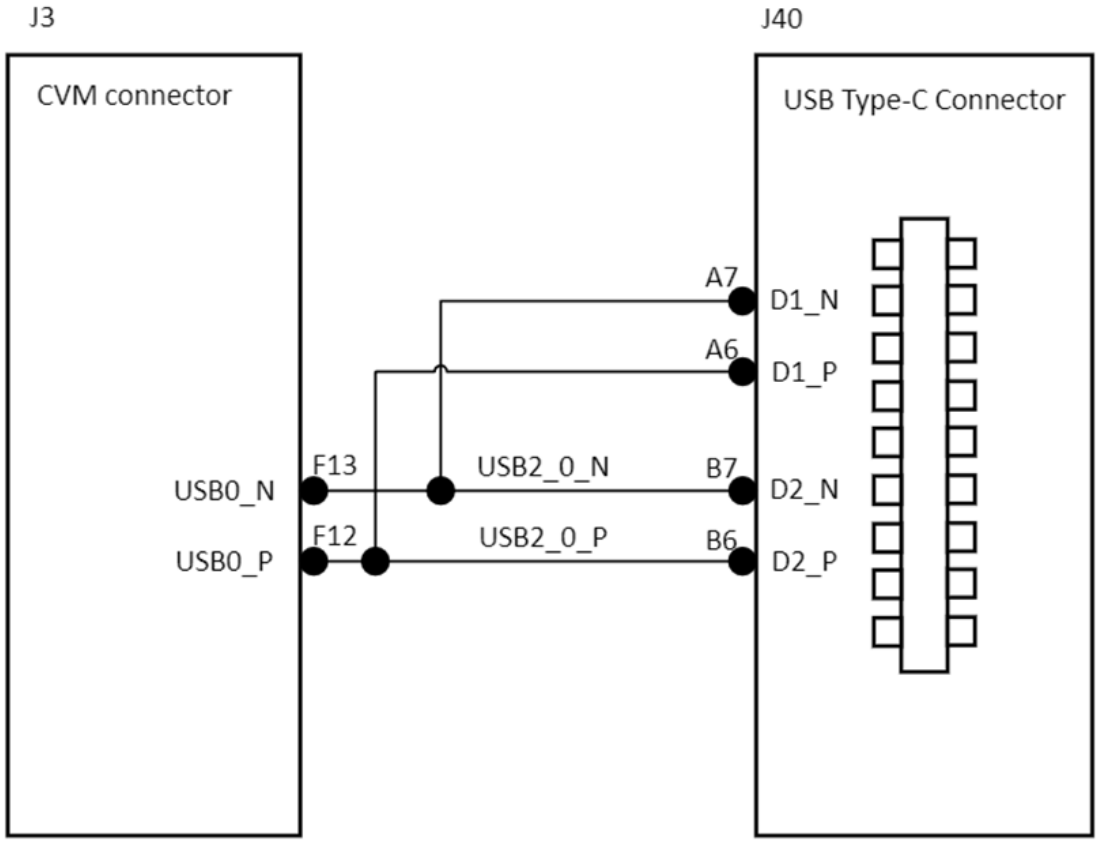
Review the Schematics

Note: **The P3737 carrier board's schematic file, P3737_A04_Concept_schematics.pdf, is included in *Jetson AGX Orin Series Developer Kit Carrier Board Design Files (A04)*, available at:**

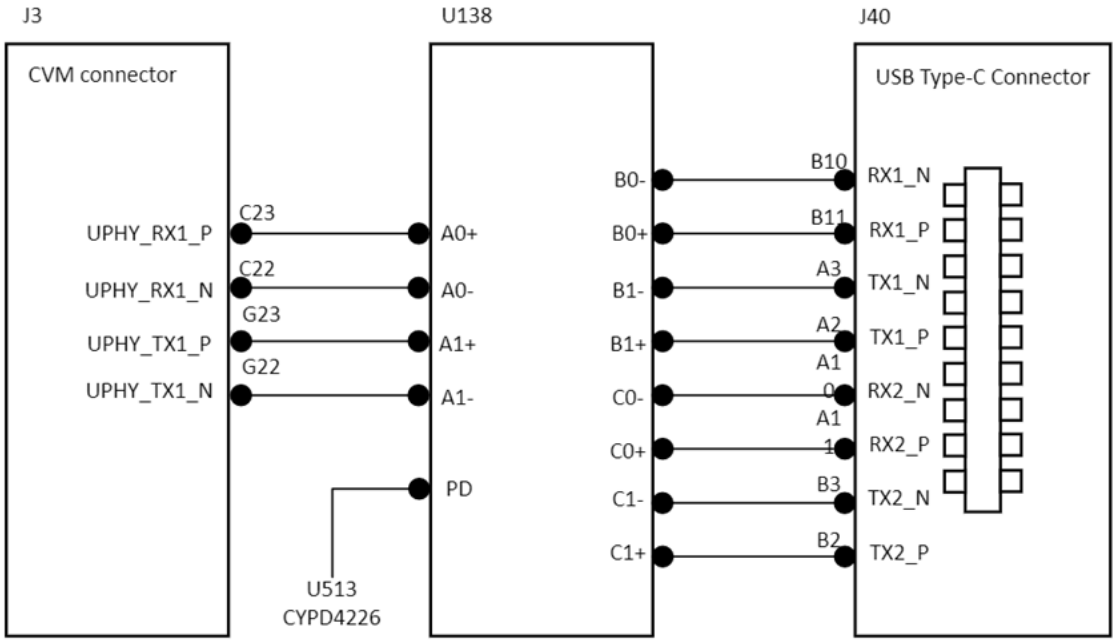
https://developer.nvidia.com/jetson-agx-orin-developer-kit-carrier-board-design-files-a04.

Check the USB connectors on the P3737 carrier board and find the wired socket location to P3701.

- USB2.0 signal pins D+/D− (USB0_*) wire out from J40 and lead to F12 (USB0_P) and F13 (USB0_N) on the SOM socket.



- USB3.2 differential signal pairs (TX* and RX*) wire out from J40 and lead to G22 (UPHY_RX1_N), G23 (UPHY_RX1_P), C22 (UPHY_TX1_N), and C23 (UPHY_TX1_P) on the SOM socket through U138 and U513, the USB type-C alt mode switch.



Through the schematic, we can see that for J40:

- The USB2.0 signal pair is wired to UTMI pad 0 (USB2 port 0).
- USB3.2 signal pairs are wired to UPHY lane 1 (USB3.2 port 1 according to UPHY lane mapping).

The USB Connector Class

A USB connector class represents a physical USB connector. It should be a child of a USB interface controller or a separate node when it is attached to the MUX and USB interface controllers.

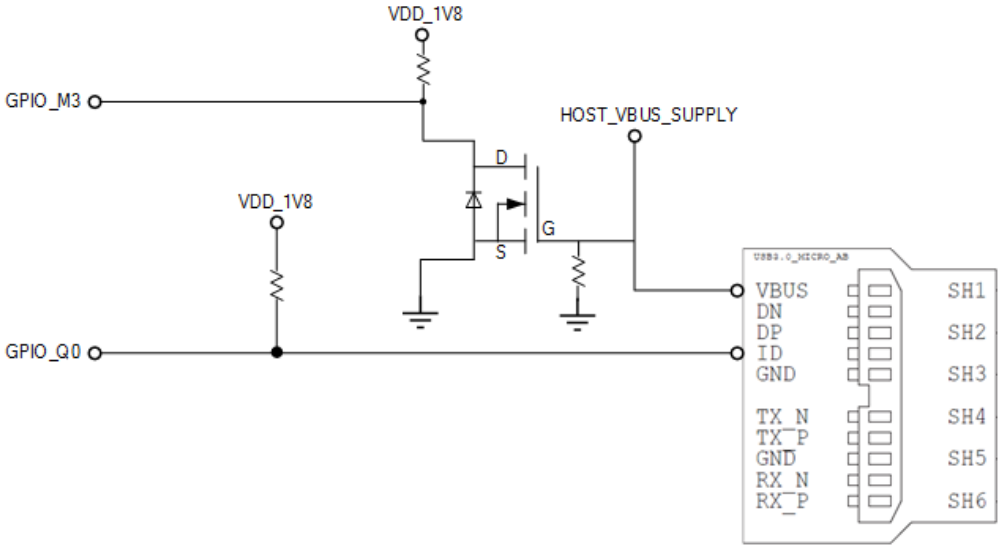
Generally, port switching between the roles of an OTG port is controlled by the host driver (xHCI) and device driver (xUDC), and can be defined by the state of the ID pin and the VBUS_DETECT pin.

Taking GPIO_M3 as the VBUS_DETECT pin and GPIO_Q0 as the ID pin, for example:

1. Find the corresponding GPIO states on the VBUS_DETECT pin and ID pin.

   Generally, the ID pin is designed as an internal pull high (logical high). With an A-plug connected the ID pin is pulled to ground (logical low), while with a B-plug connected or no cable connected it remains logical high.

   The operation of the VBUS_DETECT pin depends on the device's design. Consider the schematic in the following diagram, for example:



With a B-plug connected, VBUS_DETECT is logical low, because VBUS is provided from an external power supply, and when no cable is connected it is logical high.

Note: **VBUS_DETECT is initially logical high, then logical low because VBUS is provided by the host controller. Therefore, the state of the VBUS_DETECT pin does not matter when the OTG port is operating in host mode.**

2. Create the table of GPIO states and their corresponding output cable states:

GPIO_Q0 (ID)	GPIO_M3 (VBUS_DETECT)	Data Role
1	1	Not Connected
0	0	HOST
0	1	HOST
1	0	DEVICE

Under the Connector Node (Not Used on the P3737 Carrier Board)

Port switching between the roles of an OTG port is defined by the state of the ID pin and the VBUS_DETECT pin and the settings of the external connector class.

- `compatible`: Value must be `gpio-usb-b-connector`.
- `label`: Symbolic name for the connector
- `type`: Size of the connector, should be specified in case of non-full size 'usb-a-connector' or 'usb-b-connector' compatible connectors.
- `id-gpios`: An input gpio for USB ID pin.

- `vbus-gpios:` An input gpio for the USB VBus pin, used to detect the presence of VBUS 5V.
- `cable-connected-on-boot`: Name of the output cable connected on boot, Should be one of the “USB_ROLE_NONE”, “USB_ROLE_HOST”, and “USB_ROLE_DEVICE”. If not specified, the system assumes that no cable is to be connected.
- `wakeup-source`: A Boolean; `true` if the device can wake up the system.

For the detailed information about `USBConnectorClass`, refer to the documentation at:

```
kernel/kernel-5.10/Documentation/devicetree/bindings/connector/usb-connector.ya
ml
```

Note: **OTG port switching between the host driver (xHCI) and device driver (xUDC) roles are controlled by the Cypress Type-C controller. Therefore, this section is not a part of the device-tree for the Jetson AGX Orin Developer Kit.**

- Create an `USBConnectorClass` device node and property list based on the device tree structure described above and the table of GPIO states and corresponding output cable states for GPIO_Q0 and GPIO_M3:

```
xusb_padctl: xusb_padctl@3520000 {

        ...
        ports {
                usb2-0 {
                        ...
                        Connector {
                          compatible = "gpio-usb-b-connector";
                          label = "micro-USB";
                          type = "micro";
                          vbus-gpio = <&tegra_main_gpio
                        TEGRA194_MAIN_GPIO(M, 3) GPIO_ACTIVE_LOW>;
                          id-gpio = <&tegra_main_gpio
                        TEGRA194_MAIN_GPIO(Q, 0) GPIO_ACTIVE_HIGH>;
                        };
                        ...
                };
        };

        ...

};
```

For the example of `extcon`, refer to the device tree's source code at:

```
hardware/nvidia/platform/t19x/galen/kernel-dts/common/tegra194-e3366-1199-a0
0.dtsi"
```

Note: **Check the pinmux table for the GPIO that corresponds to the ID pin and VBUS_DETECT pin.**

Under the ucsi_ccg Node

In the Jetson AGX Orin Developer Kit role switching of port J40 between host driver (xHCI) and device driver (xUDC) modes is controlled by default by U513, a Cypress Type-C controller, and the `ucsi_ccg` driver.

- `compatible`: Value must be "`nvidia,ccgx-ucsi`".
- `ccgx,firmware-build`: The ccg firmware builder.
- `reg`: The i2c slave address of typec port controller device.
- `interrupt-parent`: The phandle to the interrupt controller which provides the interrupt.
- `interrupts`: The interrupt specification for CCGx's notification.
- `connector`: The "usb-c-connector" attached to the CCGx chip, the bindings of the connector node are specified in: `Documentation/devicetree/bindings/connector/usb-connector.yaml`.

For the detailed information about ucsi_ccg, refer to the driver source code at:

`kernel/kernel-5.10/driver/usb/typec/ucsi/ucsi_ccg.c`

Taking J40 USB3.2 type-C connector as an example, create a `ucsi_ccg` node and property list based on the device tree structure described above for J40:

```
ucsi_ccg: ucsi_ccg@8 {
   status = "okay";
   compatible = "nvidia,ccgx-ucsi";
   ccgx,firmware-build = "gn";
   reg = <0x08>;
   interrupt-parent = <&tegra_aon_gpio>;
   interrupts = <TEGRA234_AON_GPIO(BB, 2) IRQ_TYPE_LEVEL_LOW>;
   ccg_typec_con0: connector@0 {
         compatible = "usb-c-connector";
       label = "USB-C";
       data-role = "host";
   };
   ccg_typec_con1: connector@1 {
         compatible = "usb-c-connector";
       label = "USB-C";
       data-role = "dual";
       port {
         ucsi_ccg_p1: endpoint {
               remote-endpoint = <&usb_role_switch0>;
           };
       };
   };
};
```

Under the xusb_padctl Node

`xusb_padctl` settings for an OTG port are the same as for a host-only port except that the mode should be `otg`, the `usb-role-switch` property is added, and the remote endpoint settings are attached to the CCGx chip, and the bindings of connector node are specified in `Documentation/devicetree/bindings/connector/usb-connector.yaml` file.

Take J40, the USB3.2 type-C connector as an example, and create a pad/port node and property list:

```
xusb_padctl: xusb_padctl@3520000 {
   ...
   pads {
         usb2 {
               lanes {
                     usb2-0 {
                           nvidia,function = "xusb";
                           status = "okay";
                     };
                     ...
               };
         };
         usb3 {
               lanes {
                     ...
                     usb3-1 {
                           nvidia,function = "xusb";
                           status = "okay";
                     };
                     ...
               };
         };
   };
   ports {
         usb2-0 {
               mode = "otg";
               usb-role-switch;
               vbus-supply = <&battery_reg>;
               status = "okay";
               port {
                     usb_role_switch0: endpoint {
                     remote-endpoint = <&ucsi_ccg_p1>;
                     };
               };
         };
         ...
         usb3-1 {
               nvidia,usb2-companion = <0>;
               status = "okay";
         };
         ...
   };
};
```

Under the xHCI Node

The xHCI settings for an OTG port are the same as for a host-only port.

Take J40, the USB3.1 type-C connector as an example, and create an xHCI node and property list based on the device tree structure described in Under the XHCI Node for a host-only port:

```
tegra_xhci: xhci@3610000 {
    ...
    phys = <&{/xusb_padctl@3520000/pads/usb2/lanes/usb2-0}>,
           <&{/xusb_padctl@3520000/pads/usb3/lanes/usb3-1}>;
    phy-names = "usb2-0", "usb3-1";
    nvidia,xusb-padctl = <&xusb_padctl>;
    status = "okay";
    ...
};
```

Under the xUDC Node

The Jetson AGX Orin xUDC controller supports both USB 2.0 HighSpeed/FullSpeed and USB 3.1 SuperSpeed protocols.

- `charger-detector`: USB charger detection support.

  This protocol must be the phandle of the USB charger detection driver DT node.
- `phys`: An array; must contain a pointer to the node that defines each PHY in `phy-names`.
- `phy-names`: Must include an entry for each PHY used by the controller.

  The names must be of the form `<type>-<port_number>`, where `<type>` is `"usb2"` or `"usb3"`.
- `nvidia,xusb-padctl`: Must be a pointer to the `xusb-padctl` node.

  Refer to the following documentation for more information about xUCD:

  ```
  kernel/kernel-5.10
  /Documentation/devicetree/bindings/usb/nvidia,tegra-xudc.txt
  ```

Taking J40, the USB3.1 type-C connector, as an example, create an xUDC node and property list for J40 based on the device tree structure described above:

```
tegra_xudc: xudc@3550000 {
    phys = <&{/xusb_padctl@3520000/pads/usb2/lanes/usb2-0}>,
           <&{/xusb_padctl@3520000/pads/usb3/lanes/usb3-1}>;
    phy-names = "usb2-0", "usb3-1";
    nvidia,xusb-padctl = <&xusb_padctl>;
    status = "okay";
};
```

Note:	**Before designing your custom board, verify the lane mapping by consulting the Jetson AGX Orin OEM Product Design Guide.**

Change Display Function from DP to HDMI

The Jetson AGX Orin Developer Kit provides DP as the default display option. Customization is required to enable HDMI on a custom carrier board. Refer to Display Configuration and Bring-up for more information.

Program the nvethernet Driver/DT to Allow it Work with the Third-Party PHY/Switch

For PHY

Prerequisites:

- For 10 G PHYs make sure the PHY firmware is flashed in XFI 10G mode so that we can have 10G/5G/2.5G/1G/100M connections at PHY line side.
- Make sure that PHY comes out of reset and be in a ready state (read PHY id using mdio tool to make sure PHY firmware is loaded successfully) when PHY reset GPIO is toggled.
- Make sure the PHY driver is available.

Here are the DT entries that are necessary to connect the 10G MAC with the third-party Ethernet PHY:

```
/* 0:XFI 10G, 1:XFI 5G, 2:USXGMII 10G, 3:USXGMII 5G */
nvidia,phy-iface-mode = <0>;
nvidia,phy-reset-gpio = <&tegra_main_gpio TEGRA234_MAIN_GPIO(Y, 1) 0>;

mdio {
        compatible = "nvidia,eqos-mdio";
        #address-cells = <1>;
        #size-cells = <0>;

        mgbe0_aqr113c_phy: ethernet_phy@0 {
                compatible = "ethernet-phy-ieee802.3-c45";
                reg = <0x0>;
                nvidia,phy-rst-pdelay-msec = <150>; /* msec */
                nvidia,phy-rst-duration-usec = <221000>; /* usec */
        };
};
```

- `nvidia,phy-iface-mode`: 0 for XFI 10G mode, 1 for XFI 5G, 2 for USXGMII, 3 for USXGMII 5G. Default configuration is XFI 10G mode.
- Add `nvidia,uphy-gbe-mode` = 1 or 0 (1 : 10G, 0 : 5G) based on the phy-iface-mode selection. Default configuration is 10G.
- `reg`: Make sure this needs to be the MDIO address. Default MDIO address is 0.
- `nvidia,phy-rst-pdelay-msec`: Post delay value once after bringing PHY out of reset in milliseconds
- `nvidia,phy-rst-duration-usec`: Delay between the PHY reset GPIO toggle in microseconds

For Switch

Prerequisites

1. Ensure that the Switch port is configured for XFI 5G or XFI 10G.
2. Ensure that **before** Orin boots up, the Switch is powered on and that the port is configured.

```
/* 1:10G, 0:5G */
nvidia,uphy-gbe-mode = <1>;
/* 0:XFI 10G, 1:XFI 5G, 2:USXGMII 10G, 3:USXGMII 5G */
nvidia,phy-iface-mode = <0>;
nvidia,max-platform-mtu = <16383>;

fixed-link {
        speed = <10000>;
        full-duplex;
};
```

Here are the DT entries that are necessary to connect the 10G MAC with the third-party Ethernet switch:

- `nvidia,uphy-gbe-mode`: 1/0 based on the switch port configuration
- `nvidia,phy-iface-mode`: 0/1 based on the switch port configuration.
- Add the fixed-link node with speed 10G/5G based on the switch port configuration.

For RGMII

Prerequisites:

- Ensure that PHY comes out of reset mode and is in a ready state when the PHY reset GPIO is toggled.

  The read PHY ID should use the mdio tool to ensure that PHY is out of the reset mode.

- Ensure that the PHY driver is available.
- Ensure that the PIN mux settings are correct in the RGMII interface.

- Ensure that PHY comes out of reset and supplies the Rx clock to EQOS.

Here are the DT entries that are necessary to connect the 1G MAC to the third-party ethernet PHY:pinmux

```
    phy-mode = "rgmii-id";
    phy-handle = <&phy>;
    nvidia,phy-reset-gpio = <&tegra_main_gpio TEGRA234_MAIN_GPIO(G, 5) 0>;

    mdio {
        compatible = "nvidia,eqos-mdio";
        #address-cells = <1>;
        #size-cells = <0>;

        phy: phy@1 {
            reg = <1>;
            nvidia,phy-rst-pdelay-msec = <224>; /* msec */
            nvidia,phy-rst-duration-usec = <10000>; /* usec */
            interrupt-parent = <&tegra_main_gpio>;
            interrupts = <TEGRA234_MAIN_GPIO(G, 4) IRQ_TYPE_LEVEL_LOW>;

        };
    };
};
```

Here is some additional information about the DT entries:

- Ensure that the `reg` variable is set with the MDIO address.

  The default MDIO address is 1.

- `phy-mode`: phy mode (for example, rgmii-id).
- `interrupts`: PHY interrupt.
- `nvidia,phy-reset-gpio`: PHY reset GPIO
- `nvidia,phy-rst-pdelay-msec`: Post delay value after bringing PHY out of reset in milliseconds.
- `nvidia,phy-rst-duration-usec`: Delay between the PHY reset GPIO toggle in microseconds.
- Pinmux setting changes: Below are the pinmux changes that need to be taken care to configure RGMII. Need to modify all the below mentioned Tx, RX parameters with the sample settings pasted.
  - Tx
    - eqos_txc_pe0, eqos_td0_pe1, eqos_td1_pe2, eqos_td2_pe3, eqos_td3_pe4, eqos_tx_ctl_pe5

```
eqos_td0_pe1 {
        nvidia,pins = "eqos_td0_pe1";
        nvidia,function = "eqos";
        nvidia,pull = <TEGRA_PIN_PULL_NONE>;
        nvidia,tristate = <TEGRA_PIN_DISABLE>;
        nvidia,enable-input = <TEGRA_PIN_DISABLE>;
};
```

- Rx
  - eqos_rd0_pe6, eqos_rd1_pe7, eqos_rd2_pf0, eqos_rd3_pf1, eqos_rx_ctl_pf2, eqos_rxc_pf3

```
eqos_rd0_pe6 {
        nvidia,pins = "eqos_rd0_pe6";
        nvidia,function = "eqos";
        nvidia,pull = <TEGRA_PIN_PULL_UP>;
        nvidia,tristate = <TEGRA_PIN_ENABLE>;
        nvidia,enable-input = <TEGRA_PIN_ENABLE>;
};
```

- MDC/MDIO
  - eqos_sma_mdio_pf4, eqos_sma_mdc_pf5

```
eqos_sma_mdio_pf4 {
        nvidia,pins = "eqos_sma_mdio_pf4";
        nvidia,function = "eqos";
        nvidia,pull = <TEGRA_PIN_PULL_NONE>;
        nvidia,tristate = <TEGRA_PIN_DISABLE>;
        nvidia,enable-input = <TEGRA_PIN_ENABLE>;
};

eqos_sma_mdc_pf5 {
        nvidia,pins = "eqos_sma_mdc_pf5";
        nvidia,function = "eqos";
        nvidia,pull = <TEGRA_PIN_PULL_NONE>;
        nvidia,tristate = <TEGRA_PIN_DISABLE>;
        nvidia,enable-input = <TEGRA_PIN_DISABLE>;
};
```

- Reset, Interrupt Pinmux settings.

```
soc_gpio17_pg4 {
        nvidia,pins = "soc_gpio17_pg4";
        nvidia,function = "rsvd0";
        nvidia,pull = <TEGRA_PIN_PULL_UP>;
        nvidia,tristate = <TEGRA_PIN_ENABLE>;
        nvidia,enable-input = <TEGRA_PIN_ENABLE>;
        nvidia,lpdr = <TEGRA_PIN_DISABLE>;
};

soc_gpio18_pg5 {
        nvidia,pins = "soc_gpio18_pg5";
        nvidia,function = "rsvd0";
        nvidia,pull = <TEGRA_PIN_PULL_NONE>;
        nvidia,tristate = <TEGRA_PIN_DISABLE>;
        nvidia,enable-input = <TEGRA_PIN_DISABLE>;
        nvidia,lpdr = <TEGRA_PIN_DISABLE>;
};
```

UPHY Lane Configuration

In Table 6-4 in *USB 3.2, PCIe, UFS, and MGBE Mapping Options* (Jetson_AGX_Orin_Design_Guide_DG-10653-001_v1.0.pdf), there are two supported UPHY configurations for Orin. No mixing or matching of interfaces between the configurations is supported.

UPHY0/1/2 are named as HSIO, NVHS, and GBE respectively.

To select Configuration #1:

```
ODMDATA="gbe-uphy-config-22,nvhs-uphy-config-0,hsio-uphy-config-0,gbe0
-enable-10g,hsstp-lane-map-3";
```

By default, AGX Orin supports the configuration #1 mentioned above. Make sure to update the **`gbe-uphy-config`** from 22 to 0 if your CVB is not using MGBE. The board will not boot without this required change.

To select Configuration #2 :
ODMDATA="gbe-uphy-config-0,hsstp-lane-map-3,hsio-uphy-config-16**,nvhs-uphy-config-0**";

The ODMDATA string is in the p3701.conf.common file in the BSP.

Also note the following information:

- The flag gbe0-enable-10g is to configure MGBE to 10G mode.
  When the property is not present, and the GbE0 controller is enabled, it is configured to 5G mode. If this property is specified, and the GbE0 controller is not enabled, BPMP-FW will raise a fatal error and prevent the system from starting.
- The `hsstp-lane-map-3` is for hsstp debug enablement by using dstream.

ODM Data for T234

31:26	25:23	22:18	17	16	15	14	13:0
HSIO UPHY Config	NVHS UPHY Config	GBE UPHY Config	GBE3 Mode	GBE2 Mode	GBE1 Mode	GBE0 Mode	Reserved
Config Number in HSIO UPHY Lane Mapping Options	Config Number in NVHS UPHY Lane Mapping Options	Config Number in GBE UPHY Lane Mapping Options	0: 5G, 1: 10G	0: 5G, 1: 10G	0: 5G, 1: 10G	0: 5G, 1: 10G	TBD
BPMP-FW DTB `/uphy/hsio-uphy-config`	BPMP-FW DTB `/uphy/nvhs-uphy-config`	BPMP-FW DTB `/uphy/gbe-uphy-config`	BPMP-FW DTB `/uphy/gbe3-enable-10g`	BPMP-FW DTB `/uphy/gbe2-enable-10g`	BPMP-FW DTB `/uphy/gbe1-enable-10g`	BPMP-FW DTB `/uphy/gbe0-enable-10g`	TBD

HSIO UPHY Lane Mapping Options

Config Number	PLL0	Lane 0	Lane 1	PLL1	Lane 2	Lane 3	PLL2	Lane 4	Lane 5	Lane 6	Lane 7	PLL3
0	Disabled	USB 3.1 P0	USB 3.1 P1	USB3 / PCIE G2	USB 3.1 P2	PCIE x1 C1	PCIE G4	PCIE x4 C4				USB3 / PCIE G2
16	USB3 / PCIE G2	PCIE x1 C0	USB 3.1 P1	PCIE G4	USB 3.1 P2	PCIE x1 C1	USB3 / PCIE G2	PCIE x4 C4				PCIE G4

NVHS UPHY Lane Mapping Options

Config Number	PLL0	Lane 0	Lane 1	Lane 2	Lane 3	Lane 4	Lane 5	Lane 6	Lane 7	PLL1
0		PCIE x8 C5								

GBE UPHY Lane Mapping Options

Config Number	PLL0	Lane 0	Lane 1	Lane 2	Lane 3	PLL 1	Lane 4	Lane 5	Lane 6	Lane 7	PLL2
0	Disabled	PCIE x8 C7				PCIE C7 Only	PCIE x8 C7				Disabled
22	Disabled	PCIE x4 C7				PCIE C7 Only	MGBE0	MGBE1	MGBE2	MGBE3	GBE

Flashing the Build Image

When flashing the build image, use your specific board name. The flashing script uses the configuration in the `<board>.conf` file during the flashing process.

Setting Optional Environment Variables

`flash.sh` script updates the following environment variables based on board EEPROM and other parameters that were passed. To provide specific values to these variables, define them in the board-specific file `board.conf` file to override the default values.

```
# Optional Environment Variables:
# BCTFILE ---------------- Boot control table configuration file to be used.
# BOARDID ---------------- Pass boardid to override EEPROM value
# BOARDREV --------------- Pass board_revision to override EEPROM value
# BOARDSKU --------------- Pass board_sku to override EEPROM value
# BOOTLOADER ------------- Bootloader binary to be flashed
# BOOTPARTLIMIT ---------- GPT data limit. (== Max BCT size + PPT size)
# BOOTPARTSIZE ----------- Total eMMC HW boot partition size.
# CFGFILE ---------------- Partition table configuration file to be used.
# CMDLINE ---------------- Target cmdline. See help for more information.
# DEVSECTSIZE ------------ Device Sector size. (default = 512Byte).
# DTBFILE ---------------- Device Tree file to be used.
# EMMCSIZE --------------- Size of target device eMMC (boot0+boot1+user).
# FLASHAPP --------------- Flash application running in host machine.
# FLASHER ---------------- Flash server running in target machine.
# INITRD ----------------- Initrd image file to be flashed.
# KERNEL_IMAGE ----------- Linux kernel zImage file to be flashed.
# MTS -------------------- MTS file name such as mts_si.
```

```
# MTSPREBOOT ------------- MTS preboot file name such as mts_preboot_si.
# NFSARGS ---------------- Static Network assignments.
#                          <C-ipa>:<S-ipa>:<G-ipa>:<netmask>
# NFSROOT ---------------- NFSROOT i.e. <my IP addr>:/exported/rootfs_dir.
# ODMDATA ---------------- Odmdata to be used.
# PKCKEY ----------------- RSA key file to use to sign bootloader images.
# ROOTFSSIZE ------------- Linux RootFS size (internal emmc/nand only).
# ROOTFS_DIR ------------- Linux RootFS directory name.
# SBKKEY ----------------- SBK key file to use to encrypt bootloader images.
# SCEFILE ---------------- SCE firmware file such as camera-rtcpu-sce.img.
# SPEFILE ---------------- SPE firmware file path such as
bootloader/spe.bin.
# FAB -------------------- Target board's FAB ID.
# TEGRABOOT -------------- lowerlayer bootloader such as nvtboot.bin.
# WB0BOOT ---------------- Warmboot code such as nvtbootwb0.bin
```

Note: **The parameters must be added under the reference to the `<xxx>.conf.common` file to be reflected in the flashed image.**

Here is an example of environment variable settings:

```
source "${LDK_DIR}/p3701.conf.common";

PINMUX_CONFIG="tegra234-mb1-bct-pinmux-p3701-0000.dtsi";
BPFDTB_FILE=tegra234-bpmp-3701-0000-3737-0000.dtb;
DTB_FILE=tegra234-p3701-0000-p3737-0000.dtb;
TBCDTB_FILE=tegra234-p3701-0000-p3737-0000.dtb;
EMMC_BCT=tegra234-p3701-0000-p3737-0000-TE990M-sdram.dts;


MISC_CONFIG=tegra234-mb1-bct-misc-p3701-0000.dts;
```

Flashing the Build Image

Run the following command:

```
$ sudo ./flash.sh <board> mmcblk0p1
```

For more flashing support and options, refer to Flashing Support.

Users might encounter flashing/booting issues on a custom carrier board without EEPROM if the required MB2 BCT changes are not made. Refer to MB2 Configuration Changes for more information.

Note: **UEFI picks the kernel Image and dtb from the rootfs path mentioned in `/boot/extlinux/extlinux.conf` file. If you mentioned an image and dtb in this file, these items will be given precedence. For example, you might want to scp the file to the Jetson target path mentioned in the file. If this information is not mentioned, or file is not present, the Kernel Image or dtb will be selected from the flashed partition in emmc.**

Enable the eMMC EUDA

eMMC devices that are subjected to heavy loads might reach end of life (EOL) quite early in the deployment stage. For such aggressive buffering, we recommend that you enable an enhanced user data area (EUDA) on the area in the device that is configured as a Single-level cell (SLC). EUDA can be enabled using the open source `mmc_utils` tool with the following command:

```
mmc enh_area set -y <start KiB> <length KiB> /dev/mmcblk0
```

- **Example:**
  ```
  mmc enh_area set -y 0 7364608 /dev/mmcblk0
  ```
- **Expected output:**
  ```
  sudo mmc enh_area set -y 0 7364608 /dev/mmcblk0
  ```
- Enhanced User Data Area Size [ENH_SIZE_MULT]: 0x000383 (for example, 7364608 KiB)
- Max Enhanced Area Size [MAX_ENH_SIZE_MULT]: 0x000383nn (for example, 7364608 KiB)

To enable the EUDA feature:

1. Set the ENH_USR area on `/dev/mmcblk0`.
2. Set OTP PARTITION_SETTING_COMPLETED.
3. Set OTP PARTITION_SETTING_COMPLETED on /dev/mmcblk0.
4. Power cycle the device power cycle for the changes to take effect.
5. After the power cycle, confirm that PARTITION_SETTING_COMPLETED is set by using `extcsd read`.

Power cycle the device and confirm whether the PARTITION_SETTING_COMPLETED field is set.

Note:

- **Enabling EUDA reduces the overall disk size, but it is an irreversible operation.**
  Reach out to the corresponding eMMC part vendor before you enalbe EUDA. EUDA should be enabled only based on the instructions provided by the eMMC part vendor.
- **After EUDA is enabled, you cannot reflash the device because the partition table needs to be modified to work with new eMMC capacity.**

EMMC Lifecycle and Data Retention/Refresh

Device Health/End Of Life

With the Jetpack release, the customer can use the open-source `mmc_utils` tool to read relevant fields in the EXT_CSD register and estimate the device health/EOL.
The user can also dump the following debugfs nodes to monitor the device health:

```
/sys/kernel/debug/mmc0/mmc0\:0001/dhs_type_a
/sys/kernel/debug/mmc0/mmc0\:0001/dhs_type_b
/sys/kernel/debug/mmc0/mmc0\:0001/eol_status
```

Retention/Refresh

eMMCs (SSD devices) usually handle refresh internally as long as they are powered on and have requests/transfers to them. If the device is powered off for 6 months or more without any activity, data might be lost or corrupted.
If the datasheet for the part includes vendor-specific commands to refresh the device, the user can use the standard `ioctl` interface, which is similar to open source `mmc_utils`, to send required commands to the device.

Notice

This document is provided for information purposes only and shall not be regarded as a warranty of a certain functionality, condition, or quality of a product. NVIDIA Corporation (“NVIDIA”) makes no representations or warranties, expressed or implied, as to the accuracy or completeness of the information contained in this document and assumes no responsibility for any errors contained herein. NVIDIA shall have no liability for the consequences or use of such information or for any infringement of patents or other rights of third parties that may result from its use. This document is not a commitment to develop, release, or deliver any Material (defined below), code, or functionality.

NVIDIA reserves the right to make corrections, modifications, enhancements, improvements, and any other changes to this document, at any time without notice.

Customer should obtain the latest relevant information before placing orders and should verify that such information is current and complete.

NVIDIA products are sold subject to the NVIDIA standard terms and conditions of sale supplied at the time of order acknowledgement, unless otherwise agreed in an individual sales agreement signed by authorized representatives of NVIDIA and customer (“Terms of Sale”). NVIDIA hereby expressly objects to applying any customer general terms and conditions with regards to the purchase of the NVIDIA product referenced in this document. No contractual obligations are formed either directly or indirectly by this document.

NVIDIA products are not designed, authorized, or warranted to be suitable for use in medical, military, aircraft, space, or life support equipment, nor in applications where failure or malfunction of the NVIDIA product can reasonably be expected to result in personal injury, death, or property or environmental damage. NVIDIA accepts no liability for inclusion and/or use of NVIDIA products in such equipment or applications and therefore such inclusion and/or use is at customer’s own risk.

NVIDIA makes no representation or warranty that products based on this document will be suitable for any specified use. Testing of all parameters of each product is not necessarily performed by NVIDIA. It is customer’s sole responsibility to evaluate and determine the applicability of any information contained in this document, ensure the product is suitable and fit for the application planned by customer, and perform the necessary testing for the application in order to avoid a default of the application or the product. Weaknesses in customer’s product designs may affect the quality and reliability of the NVIDIA product and may result in additional or different conditions and/or requirements beyond those contained in this document. NVIDIA accepts no liability related to any default, damage, costs, or problem which may be based on or attributable to: (i) the use of the NVIDIA product in any manner that is contrary to this document or (ii) customer product designs.

No license, either expressed or implied, is granted under any NVIDIA patent right, copyright, or other NVIDIA intellectual property right under this document. Information published by NVIDIA regarding third-party products or services does not constitute a license from NVIDIA to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property rights of the third party, or a license from NVIDIA under the patents or other intellectual property rights of NVIDIA.

Reproduction of information in this document is permissible only if approved in advance by NVIDIA in writing, reproduced without alteration and in full compliance with all applicable export laws and regulations, and accompanied by all associated conditions, limitations, and notices.

THIS DOCUMENT AND ALL NVIDIA DESIGN SPECIFICATIONS, REFERENCE BOARDS, FILES, DRAWINGS, DIAGNOSTICS, LISTS, AND OTHER DOCUMENTS (TOGETHER AND SEPARATELY, “MATERIALS”) ARE BEING PROVIDED “AS IS.” NVIDIA MAKES NO WARRANTIES, EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE MATERIALS, AND EXPRESSLY DISCLAIMS ALL IMPLIED WARRANTIES OF NONINFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE. TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL NVIDIA BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF NVIDIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Notwithstanding any damages that customer might incur for any reason whatsoever, NVIDIA’s aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms of Sale for the product.

NVIDIA Corporation | 2788 San Tomas Expressway, Santa Clara, CA 95051
http://www.nvidia.com

NVIDIA Corporation | 2788 San Tomas Expressway, Santa Clara, CA 95051
http://www.nvidia.com