PLATFORM ADAPTATION AND BRING-UP GUIDE

DA_08477-001 | March 8, 2018

Tegra Linux Driver Package for Jetson TX2

Document Change History

DA_08477-001

Version	Date	Authors	Description of Change
v1.0	2 Mar 2017	twarren/bbasu/snath	Initial release for Jetson TX2
v1.1	14 Mar 2017	bbasu	Added Power Tree changes
v1.2	8 May 2017	bbasu	board configuration updates
v1.3	30 Jun 2017	jerchang/mzensius	GPIO-related updates
v1.4	29 Jan 2018	dliu/kstone	Device tree settings for QSPI_IO2

Table of Contents

Platform Adaptation and Bring-Up Guide

This document describes how to port the NVIDIA® Tegra® Linux Driver Package and the U-Boot bootloader from NVIDIA® Jetson™ TX2 Developer Kit to other hardware platforms.

The examples described include code for the Jetson TX2 Developer Kit (P2771).

For information on customizing the configuration files, refer to the *Tegra Linux Driver Package Development Guide* "MB1 Platform Configuration" and "Configuring Pinmux, GPIO and PAD" topics.

Board Configuration

The Jetson TX2 module consists of a P3310 main board that sits on a P2597 base board. The complete kit is named P2771 Jetson TX2 Developer Kit. The P3310 main board can be used without any software configuration modifications. The P3310 board sits on the P2597 I/O expansion base board. Both these boards have an EEPROM where the board ID is saved.

Before replacing the P2597 base board, verify the software programming of the Kernel device tables, MB1 configuration, ODM data, and flashing to de-configure the P2597 board with the custom configurations of your custom board. EEPROM ID for your custom board is not required.

Board Naming

To support your board in L4T, you must select a simple lower-case, alpha-numeric name for your board. The name can include dashes (-) or underscores (_) but cannot contain spaces. For example:

```
jetson-tx2
p2771-000-500
myboard
```

The name you select appears in:

- Filenames and pathnames
- U-Boot and Linux kernel source code
- User-visible device tree filenames

Additionally, this name is exposed to the user through the U-Boot command prompt and various Linux kernel `proc` files.

In this document, `<board>` represents your board name.

You must also select a similarly-constructed vendor name. The same character set rules apply, such as the following example:

```
nvidia
```

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

> **Note:** **Do not re-use and modify the existing NVIDIA® Jetson™ TX2 Developer Kit code without selecting and using your own board name. If you do not use your own board name it will not be obvious to Jetson TX2 users whether the modified source code supports the original Jetson TX2 Developer Kit board or your board.**

Placeholders in the Porting Instructions

Placeholders are used throughout this document, substitute an appropriate value for each placeholder when executing commands.

- `<function>` is a functional module name, which may be `power-tree`, `pinmux`, `sdmmc-drv`, `keys`, `comm` (Wifi/BT), `camera`, etc.
- `<board>` is a name you have selected to represent your platform. For example, `p2771` is the name of the Jetson TX2 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. 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 root file system for the platform. The supported value is `emmc`.

MB1 Configuration Changes

MB1 provides the boot time configuration of the hardware applied by the bootloader. The MB1 boot configuration tables are available at:

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

Pinmux Changes

If your board schematic differs from that for Jetson™ TX2 Developer Kit board, you must change the pinmux configuration applied by the software.

The `Jetson-TX2-Generic-Customer-Pinmux-Template.xlsm` spreadsheet is provided to:

- Show the locations and default pinmux settings
- Define the pinmux settings in the source code or device tree

  The spreadsheet is available at:

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

You must customize the spreadsheet for the configuration of your board.

GPIO Changes

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

For example, to check the GPIO number of GPIO15/AP2MDM_READY. perform the following steps.

To check the GPIO number

1. Search for `GPIO15_AP2MDM_READY` in `Jetson_TX2_Generic_Customer_Pinmux_Release.xlsx`.
2. Confirm that the Customer Usage field is applied to `GPIO3_PBB.00`.
3. Confirm in tegra186-gpio.h that GPIO3_PBB.00 belongs to the main Tegra GPIO group, and that the port number is 21:

   ```
   #define TEGRA_MAIN_GPIO_PORT_BB 21
   ```

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

```
$ dmesg | grep gpiochip_add_data
[    1.247404] gpiochip_add_data: registered GPIOs 320 to 511 on
device: tegra-gpio
[    1.262595] gpiochip_add_data: registered GPIOs 256 to 319 on
device: tegra-gpio-aon
```

   As shown in the outpout above, there are 2 tegra GPIO ports with different offsets:

   - tegra-gpio, offset = 320
   - tegra-gpio-aon, offset= 256

5. Because `PBB00` belongs to the tegra-gpio group, the port number from step 3 is 21, and the offset is 320. Use the following formula to calculate the GPIO number:

```
TEGRA_MAIN_GPIO(port, offset) =
((TEGRA_MAIN_GPIO_PORT_##port * 8) + offset)
```

Hence, the GPIO number of `GPIO15/AP2MDM_READY` is (21*8)+320 = 488.

PMIC Changes

The PMIC configuration file configures the initial PMIC in the P3310 board. Some GPIO expander-based GPIO regulator settings in the P2597 base board configurations are also defined. Review this configuration file to replace any references to the P2597 board to your custom baord. If required, include any regulator information to enable this file.

For example, remove the following section that is writing to a slave on the I2C controller 0 address 0x74 in the P2597 base board. Additionally, update the number of blocks and array number for other entries of the block:

```
tegra186-mb1-bct-pmic-quill-p3310-1000-c04.cfg

# 5V0_HDMI_EN
pmic.generic.1.block[2].type = 1; # I2C Type
pmic.generic.1.block[2].i2c-controller-id = 0;
pmic.generic.1.block[2].slave-add = 0xE8; # 7BIt:0x74
pmic.generic.1.block[2].reg-data-size = 8;
pmic.generic.1.block[2].reg-add-size = 8;
pmic.generic.1.block[2].block-delay = 10;
pmic.generic.1.block[2].count = 2;
pmic.generic.1.block[2].commands[0].0x07.0xFF = 0xEF;
pmic.generic.1.block[2].commands[1].0x03.0xFF = 0x10;
```

Porting U-Boot

Perform the following actions in the U-Boot source code to add support for your board.

1. Copy `include/configs/p2771-0000.h` to `include/configs/<board>.h`.
2. Edit the set of enabled devices and features in `<board>.h` as appropriate for your board. For example, you must change the following:

```
#define CONFIG_TEGRA_BOARD_STRING       "NVIDIA p2771-0000"
```

3. Copy `arch/arm/dts/tegra186-p2771-0000-500.dts` to `arch/arm/dts/tegra186-<board>.dts`.
4. Edit the set of enabled devices and their parameters (e.g. GPIO and IRQ IDs) in `tegra186-<board>.dts` as appropriate for your board.

   You may need to add, remove, or edit nodes and properties.

   Note: **U-Boot and the Linux kernel do not always use the exact same device tree schema (bindings) to represent the same data. Follow examples from U-Boot rather than from the Linux kernel.**

5. Add `tegra186-<board>.dtb` to `arch/arm/dts/Makefile`.
6. Copy `configs/p2771-0000-500_defconfig` to `configs/<board>_defconfig`.
7. Edit `<board>_defconfig` to refer to your board name.
8. Edit `arch/arm/mach-tegra/tegra186/Kconfig` to add target config and Kconfig.
9. Copy the `board/nvidia/p2771-0000/` directory to `board/<vendor>/<board>/`.
10. Edit all the files in `board/<vendor>/<board>/` to refer to your board name rather than the `p2771-0000`. The files in this directory contain many instances of the `p2771-0000` board name.
11. Edit `board/<vendor>/<board>/MAINTAINERS` to provide the correct maintainer contact information for your board.
12. Edit `board/<vendor>/<board>/<board>.c` to provide the correct PMIC programming for your board, if required.

Porting the Linux Kernel

It is assumed that you are using the CVM module provided by NVIDIA and that it has not been modified; the eMMC, PMIC, and DDR are the same with the same routing of lines. The modifications you are making are for the CVB baseboard that hosts all the peripherals. Consequently, based on the peripherals present on your baseboard, 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 available at:

```
<top>/hardware/nvidia/platform/t18x/
<top>/hardware_nvidia/soc/t18x
```

The final DTB file used is:

```
tegra186-quill-p3310-1000-a00-00-base.dts
```

By reading the above file, you see which other `.dtsi` files are referenced by include statements. Common `.dtsi` files that may be modified to reflect hardware design changes include:

Types of Changes	DTSI Filename or location
Power supply changes	tegra186-quill-power-tree-p3310-1000-a00-00.dtsi
Regulator parameter changes	tegra186-quill-spmic-p3310-1000-a00-00.dtsi
Display panel and node changes	Refer to the *Tegra Linux Driver Package Development Guide Display Configuration and Bringup* topic for details.
ODM data based feature configuration	tegra186-odm-data-plugin-manager.dtsi
Tegra SOC controller state to enable/disable a controller	`soc/t18x/kernel-dts/tegra186-soc/`
Panels related .dts files	`tegra/common/kernel-dts/panels/`

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

As a best practice, create your own set of `.dts` files based on the Quill files already present. 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 as follows:

```
dtc -I dtb -O dts tegra186-quill-p3310-1000-a00-00-base.dtb > tegra186-
quill-p3310-1000-a00-00-base.dts
fdtdump dts tegra186-quill-p3310-1000-a00-00-base.dtb > tegra186-quill-
p3310-1000-a00-00-base.dts
```

Power Tree Changes

The Jetson P2597 baseboard has a GPIO expander. Some of the pins on the GPIO expander are used as a GPIO regulator. One such usage is to enable `vbus-2-supply` which is powered using `vdd_usb2_5v` GPIO regulator. If your custom board does not have the `vdd_usb2_5v` supply, the `xhci` driver enumeration fails on the target system. To solve this situation, you must:

1. Change the supply with `battery_reg` using the `.dtsi` file located at:

    ```
    hardware/nvidia/platform/t18x/common/kernel-dts/t18x-common-
    platforms/tegra186-quill-power-tree-p3310-1000-a00-00.dtsi
    ```

2. Regenerate the DTB.
3. Flash with the correct DTB.

The modifications are as follows:

```
pinctrl@3520000 {
    vbus-0-supply = <&vdd_usb0_5v>;
    vbus-1-supply = <&vdd_usb1_5v>;
    vbus-2-supply = <&battery_reg>;
    vbus-3-supply = <&battery_reg>;
    vddio-hsic-supply = <&battery_reg>;
    avdd_usb-supply = <&spmic_sd3>;
    vclamp_usb-supply = <&spmic_sd2>;
    avdd_pll_erefeut-supply = <&spmic_sd2>;
};
```

To disable xhci

1. Change the lane configuration.
2. Update the following node.

```
xhci@3530000 {
    status = "disabled";
    phys = <&tegra_xusb_padctl TEGRA_PADCTL_PHY_UTMI_P(0)>,
           <&tegra_xusb_padctl TEGRA_PADCTL_PHY_UTMI_P(1)>,
           <&tegra_xusb_padctl TEGRA_PADCTL_PHY_USB3_P(1)>;
    phy-names = "utmi-0", "utmi-1", "usb3-1";
};
```

For information about `.dts` files, refer to the documentation at `Documentation/devicetree/bindings` in the NVIDIA released Linux kernel source package.

USB Lane Mapping

USB 3.0 has 4 super-speed ports. Not all can be used in the same implementation because of lane sharing between PCIE, SATA, and XUSB. Possible combinations for USB 3.0 are as follows.

	Jetson TX2 Pin Names			PEX1	PEX_RFU	PEX2	USB_SS1	PEX0	USB_SS0 See Note 1	SATA
	Tegra Lanes			Lane 0	Lane 1	Lane 3	Lane 2	Lane 4		Lane 5
	Available Outputs from Jetson TX2									
Configs	USB 3.0	PCIe	SATA							
1	0	1x1 + 1x4	1	PCIe#2_0	PCIe#0_3	PCIe#0_2	PCIe#0_1	PCIe#0_0		SATA
2*	1	1x4	1		PCIe#0_3	PCIe#0_2	PCIe#0_1	PCIe#0_0	USB_SS#0	SATA
3	2	3x1	1	PCIe#2_0	USB_SS#1	PCIe#1_0	USB_SS#2	PCIe#0_0		SATA
4	3	2x1	1		USB_SS#1	PCIe#1_0	USB_SS#2	PCIe#0_0	USB_SS#0	SATA
5	1	2x1 + 1x2	1	PCIe#2_0	USB_SS#1	PCIe#1_0	PCIe#0_1	PCIe#0_0		SATA
6	2	1x1 + 1x2	1		USB_SS#1	PCIe#1_0	PCIe#0_1	PCIe#0_0	USB_SS#0	SATA
* Default Usage on Carrier Board				Unused	X4 PCIe Connector				USB 3 Type A	SATA

** Notes:

1. PCIe Interface #2 can be brought to the PEX1 pins, or USB 3.0 Port #1 to the USB_SS0 pins on Jetson TX2 depending on the setting of a multiplexor on the module. The selection is controlled by QSPI_IO2 configured as a GPIO. For more information, see Under pinctrl to pinmux node: in Required Device Tree Changes.
2. Jetson TX2 has been designed to enable usecases listed in this table. However, released Software may not support all configurations, nor has every configuration been validated.
   - Configuration #1 & 2 represent the supported and validated Jetson TX2 Developer Kit configurations. The released software supports these configurations. The PCIe, USB 3.0, and SATA interfaces have been verified on the carrier board.
3. Lane allocation can be performed by either ODMDATA, in p2771-0000.conf.common by default, or alloted to UPHY to each client in the bpmp-dtb file.
   For example:
   - ODMDATA=0x1090000 while flashing for Jetson TX2 for Configuration #2
   - ODMDATA=0x90000 for Configuration #1
   - ODMDATA=0x6090000 for Configuration #3
4. The cell colors highlight the different PCIe and USB 3.0 Ports.
   - Three shades of green are used for PCIe interfaces #[2:0].
   - Three shades of blue are used for USB 3.0 ports #[2:0].
   - SATA is highlighted in orange.

The customer pinmux spreadsheet contains all Jetson TX2 pin names and ball names to represent which ball names are used for the super-speed connector, and the pinmux configuration of those pins.

An example configuration is available in the *Jetson TX2 OEM Product Design Guide.* Each external super-speed connector has both USB 2.0 (DP, DN) and USB 3.0 lines (TX+-, RX+) linked to the connector. A possible exception is where a fixed on-board device is connected to super-speed lines and does not require USB 2.0 compatibility.

Note:	**Before designing your custom board, verify the lane mapping and**

> **compatibility between Jetson TX1 and TX2 by consulting the *Jetson TX1 OEM Product Design Guide* available at:**
>
> **http://developer.nvidia.com/embedded/dlc/jetson-tx1-oem-product-design-guide**

Required Device Tree Changes

The following device tree properties must change when the USB configuration changes. The following examples describe the changes required to use the `xhci` controller:

- One USB 2.0 and,
- One USB 3.0 port.

Under the XHCI node:

- List all UPHY lanes required, for example:

```
phys = <PADCTL_UTMI(0)>,    // for USB2.0 port0
   <PADCTL_USB3(0)>,        // for USB3.0 port0
```

- Provide a naming convention to retrieve the above UPHYs from the kernel:

```
phy-names = "utmi-0", "usb3-0";
```

Under pinctrl to pinmux node:

- Create a node for each UPHY lane, for example:
  - For usb2.0, usb2-port0 {   // node name could be anything
    - nvidia,lanes = "otg-0";   // Required properties.
    - nvidia,function = "xusb";   // Optional properties, as per padctrl documentation, function property is required for USB2.0 port and in this case, it is “xusb”
    - nvidia,port-cap = "PORT_HOST_ONLY"; // Optional properties and could be “PORT_OTG”
  - For USB3.0, usb3-port0 {
    - nvidia,lanes = "usb3-0";   // required property
    - nvidia,port-cap = "PORT_OTG_CAP"; // required property for lane usb3-0.

The above allocations will work when UPHY lanes are owned by each client (`xhci/xudc/pcie..`).

As per the above example:

- XHCI must be owned by LAN0 for `usb3-0`.

- Lane allocation can be performed by either ODMDATA or allotted to UPHY to each client in the `bpmp-dtb` file.
- ODM Data: Bit 24 to 28 are used to allocate lanes with Lane3, by default, is allocated to PCIE.
- ODMDATA=0x1090000; While flashing for Jetson TX2.

Lane allocations are as follows:

28	27	26	25	24
USB PHY Lane5	USB PHY Lane4	USB PHY Lane2	USB PHY Lane1	USB PHY Lane0

For the detailed information about UPHY lanes, refer to the documentation at:

```
kernel/t18x/Documentation/devicetree/bindings/pinctrl/nvidia,tegra186-
padctl.txt
```

- To switch USB_SS0 to PEX1, configure QSPI_IO2 as follows:

```
pcie0_lane2_mux {
    gpio-hog;
    gpios = <TEGRA_MAIN_GPIO(R, 3) 0>;
    output-low;
    label = "pcie-lane2-mux";
-     status = "disabled";
+     status = "okay";
};
```

Flashing the Build Image

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

To flash the build image

- Execute the following command.

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

Hardware Bring-Up Checklist

This section provides a checklist for the platform hardware bring-up process.

Before Power-On

Make sure that the Jetson TX2 is connected to the BTB connector correctly and securely.	☐
Verify that power supplies are not shorted to ground or to other power supplies.	☐

Initial Power-On

Verify that VDD_IN from carrier board is in the 6 V to 19 V range.	☐
Verify that CARRIER_PWR_ON goes to HIGH when power is turned on.	☐
Verify that system can enter force recovery.	☐

Initial Software Flashing

Verify that system can be flashed with TegraFlash.	☐
Verify that TegraBoot and U-boot run to completion by checking log output.	☐
Verify that OS runs to desktop.	☐
Verify that any UARTs intended for debugging are enabled and functional.	☐

Power

Verify that all supplies required on at power-on are enabled appropriately.	☐
Verify that all supplies required off at power-on are not enabled initially.	☐
Verify that each controllable supply can be enabled and disabled, and different voltage levels can be set if applicable.	☐
Verify that carrier board power-on sequence starts after CARRIER_PWR_ON signal is asserted.	☐

Power Optimization

Capture CPU_PWR_REQ entering and exiting Suspend (LP1) and Deep Sleep (LP0). Ensure that CPU_PWR_REQ and associated power rail sequence meets Tegra Data Sheet requirements.	☐
Verify that all rails which must be OFF in Deep Sleep (LP0) are OFF.	☐
Verify that all rails which must be ON in Deep Sleep (LP0) are ON.	☐
Verify that required rails are back and at correct voltage under hardware control exiting Deep Sleep (LP0).	☐

USB 2.0 PHY

Verify that USB0 supports USB Recovery (device mode).	☐
Verify that USB0 device mode works with intended peripheral types, if supported.	☐
Verify USB0, USB1 and or USB2 Host mode, if implemented.	☐
Verify USB0 Device/Host detection, if supported.	☐
Verify that USB PHYs go to lowest power mode when not used or when the system is in low power mode.	☐
Verify that AVDD_USB and AVDD_PLL_UTMIP are off during Deep Sleep (LP0).	☐
Capture USB0_D+/D- signals at both ends of link (connector and test points near Tegra).	☐
Capture USB2_D+/D- signals at both ends of link (connector and test points near Tegra).	☐
Using USB-IF procedures, verify that signals meet requirements (correct eye height/width, etc.).	☐
If USB signals do not meet requirements, use the *Tegra USB Tuning Guide* to adjust settings until requirements are met.	☐

USB 3.0

Verify USB 3.0 Host mode.	☐
Verify USB 3.0 Device mode, if enabled.	☐
Verify that the USB 3.0 interface goes to the lowest power mode when not used or when the system is in low power mode.	☐

HDMI

Verify that HDMI-compatible display works at 1080p.	☐
Verify that display is detected properly (HPD).	☐
Verify that HDMI reads and writes to the display using DDC interface.	☐
Verify that HDMI related rails are powered off when not used or system is in Deep Sleep (LP0) or Suspend (LP1).	☐
Capture HDMI signals at the connector (using appropriate test fixture and termination).	☐
Verify that signal quality is acceptable (meets EYE diagram, etc.). Consult *Tegra HDMI Tuning Guide* for details.	☐
If HDMI signals do not meet requirements, use the *Tegra HDMI Tuning Guide* to adjust settings until requirements are met.	☐

Audio

Verify reads and writes on I2C interface used for Audio Codec.	☐

Verify that playback works properly on speakers, headphones, and headset.	☐
Verify that capture works properly: Sound is recorded from microphone/headset if supported.	☐
Verify that tones, voice, etc. can be heard from speakers or headphones/headset.	☐
Verify that Audio Codec goes to lowest power mode when not in use or system enters low power mode.	☐
Capture signals at receiver end of link, if accessible, for each I2S I/FT used.	☐
Verify that signal quality is acceptable. Look for excessive over/undershoot and glitches on signal edges.	☐

UART

Verify that Tegra TX/RX/CTS/RTS connects to device RX/TX/RTS/CTS for each UART used.	☐
Verify that signal quality is acceptable. Look for excessive over/undershoot and glitches on signal edges.	☐

SD Card (SDMMC1)

Verify proper connectivity by setting Tegra pins to GPIOs, if necessary, to debug.	☐
Verify that basic SD commands operate properly.	☐
Verify reads and writes for a variety of SD Cards.	☐
Verify that SD Card insertion detection works and wakes system, if supported.	☐
Verify that SD Card Write Protect works, if implemented.	☐
Verify that SD Card goes to low power mode or rails are powered off when not used or in low power system state.	☐
Verify that signal quality is acceptable when probed at receiver end (socket or test points near BTB connector or both for bidirectional signals). Look for excessive over/undershoot and glitches on signal edges and abnormal Clock duty cycle.	☐

Sensors I2C: General

Verify that addresses of all I2C devices appear correctly, and no unknown ghost devices appear.	☐
Verify that signal quality is acceptable, including rise times of signals, when probed at BTB connector and devices.	☐

Sensors I2C: Touch Screen (Optional)

Verify that Reads/Writes on I2C or SPI to Touch Screen controller are functional (reading device ID or a similar register is successful).	☐

Verify that interrupts are generated properly.	☐
Verify functionality of Touch Screen.	☐
Verify that Touch Screen Controller goes to lowest power mode when not used, or system is in low power state.	☐

PEX (Optional)

Verify proper connectivity by checking lanes.	☐
Verify that any implemented PEX interfaces transition to the lowest power state in Deep Sleep (LP0) and Suspend (LP1).	☐
Verify that signal quality is acceptable when probed at receiver end of link near Tegra and device. Look for excessive over/ undershoot and glitches on signal edges.	☐

SATA (Optional)

Verify proper connectivity by checking diff lines.	☐
Verify that any implemented SATA interfaces transition to the lowest power state in Deep Sleep (LP0) and Suspend (LP1).	☐
Verify that signal quality is acceptable when probed at receiver end of link near Tegra and device. Look for excessive over/ undershoot and glitches on signal edges.	☐

Embedded Display(s) (Optional)

Verify that I2C or other control interface is able to perform writes/reads to display.	☐
Verify that each embedded display shows correct colors.	☐
Verify that each embedded display’s backlight is enabled when in normal display mode.	☐
Verify that each embedded display’s backlight brightness can be adjusted properly.	☐
Verify that each embedded display’s backlight is disabled when in a low power mode.	☐
Verify that each embedded display (and any display bridge) transitions to the lowest power state in Deep Sleep (LP0) and Suspend (LP1).	☐
Verify that power-on/off sequencing of rails associated with each display meets manufacturer's requirements.	☐
Verify DSI, LVDS or eDP timing (see *Tegra DC and DSI Debugging Guide* for details on how and what to verify).	☐
Probe DSI, LVDS or eDP signals near panel driver, or at connector/test points if access to driver is not possible, and verify that signal quality is acceptable. Look for excessive over/undershoot and glitches on signal edges.	☐

Imager(s) (Optional)

Verify that I2C interface writes/reads work to all cameras.	☐
Verify that preview displays properly for all cameras.	☐
Verify that still capture works on all cameras.	☐
Verify that video capture works on all cameras.	☐
Verify that cameras and related circuitry enter lowest power mode when not used or system is in a low power mode.	☐
Verify that power-on/off sequencing of rails associated with imager module meets manufacturer's requirements.	☐
Probe MCLK output at recommended test points, and verify that signal quality is acceptable. Look for excessive over/undershoot and glitches on signal edges.	☐
Look for excessive over/undershoot and glitches on signal edges.	☐

Software Bring-Up Checklist

This section provides a checklist for the software bring-up process.

Preparation

If your replaced the SDRAM MB1 BCT with a new DDR, verify it.	☐
If you replaced the baseboard, verify the PMIC and pinmux configuration.	☐
If you replaced the eMMC, verify its operation.	☐
Obtain board schematics and component data sheets.	☐
Verify power tree and modify device tree, MB1 PMIC configuration accordingly, for the base board.	☐
Review board pinmux and modify MB1 pinmux and PAD configuration, accordingly.	☐

Bring-up Hardware Validation

Power and Reset Sequence, Power Rail Check	☐
Recovery Mode	☐
NvTest (Tegra MODS) DDR, eMMC, CPU	☐
JTAG connection check	☐

U-Boot Port and Boot Validation

TegraFlash	☐

UART output	☐
KBD connection	☐
Board config/PMIC regulator config/Pinmux/Review device tree	☐
Verify FS support/Config boot scripts (bootcmd)	☐
Boot to U-boot	☐
Boot to kernel	☐
Boot to kernel command line or custom desktop	☐

Kernel and Peripherals, Port and Validation

Device tree review, Pinmux, GPIO, Wake pins	☐
PMU and regulator drivers	☐
Display/HDMI	☐
Audio codec	☐
Microphone and speaker	☐
USB	☐
SD card	☐
Thermal Sensor	☐
EMC DFS table	☐
Ethernet	☐
SATA	☐
PCIe	☐

System Power and Clocks

CPU/CORE/GPU DVFS	☐
EMC DFS table	☐
CPU/CORE EDP	☐
GPU EDP	☐
System EDP (Contain Current monitor & Voltage comparator)	☐
Power Off	☐
LP0 (optional)	☐
CPU power down	☐
BCT, Full-speed	☐

www.nvidia.com