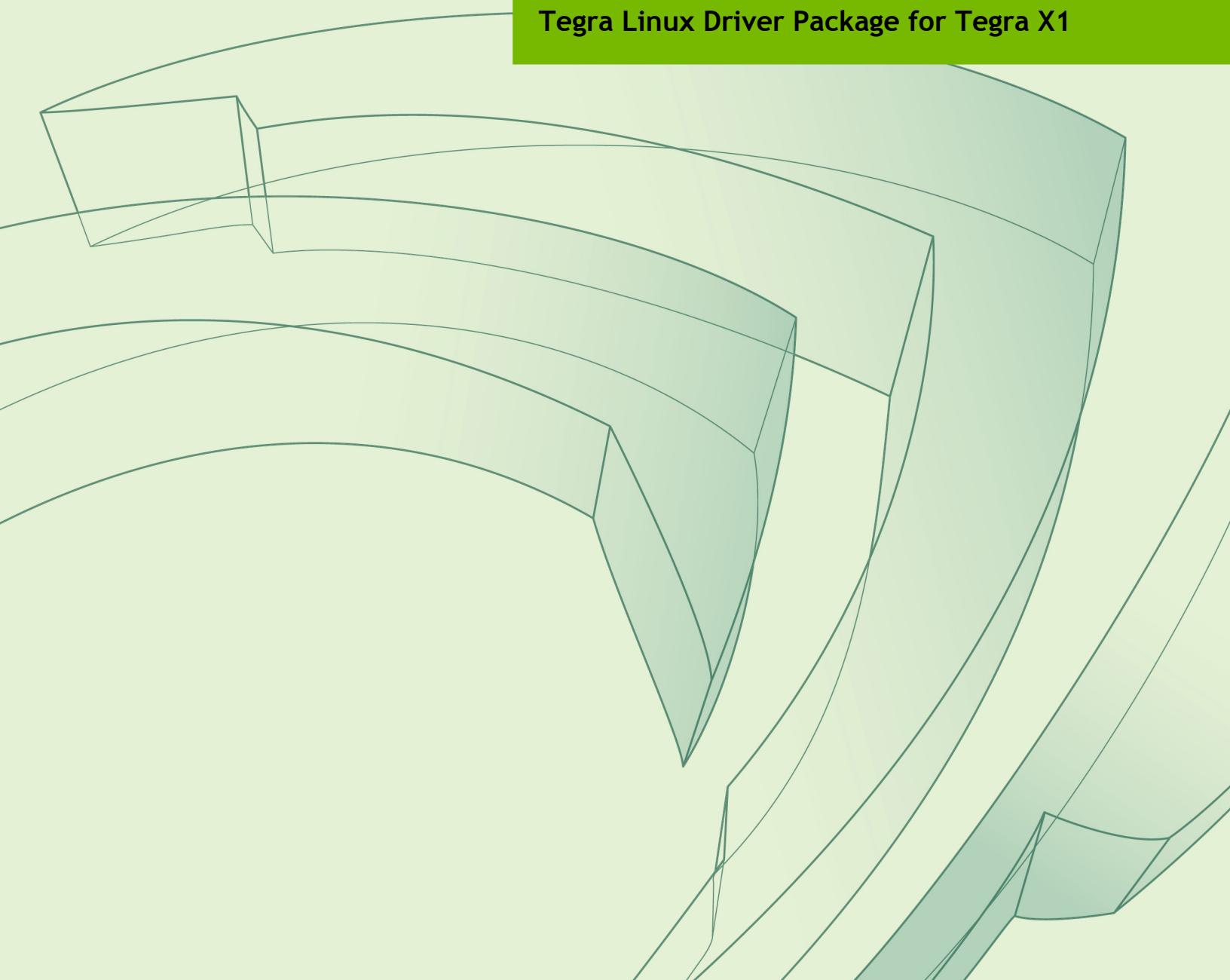




# PLATFORM ADAPTATION AND BRING-UP GUIDE

DA\_07839-001\_01 | March 1, 2016

**Tegra Linux Driver Package for Tegra X1**



## DOCUMENT CHANGE HISTORY

DA\_07378-001\_01

Version	Date	Authors	Description of Change
v1.0	1 Mar 2016	msum, jsachs	Initial release

## TABLE OF CONTENTS

<b>Platform Adaptation and Bring-Up Guide .....</b>	<b>4</b>
Porting Linux for Tegra (L4T) to Your Design .....	4
Board Naming .....	4
Placeholders in the Porting Instructions .....	5
Pinmux Changes .....	5
Exporting Pinmux for U-Boot .....	6
Exporting Pinmux for the L4T Linux Kernel .....	7
Porting U-Boot .....	8
Porting the Linux Kernel .....	9
Pad Power Detection .....	10
Other Considerations When Porting .....	12
Hardware Bring-Up Checklist .....	13
Before Power-On .....	13
Initial Power-On .....	14
Initial Software Flashing .....	14
Power .....	14
Power Optimization .....	14
USB 2.0 PHY .....	14
USB 3.0 .....	15
HDMI .....	15
Audio .....	15
UART .....	16
SD Card (SDMMC1) .....	16
Sensors I2C: General .....	16
Sensors I2C: Touch Screen (Optional) .....	16
PEX (Optional) .....	16
SATA (Optional) .....	17
Embedded Display(s) (Optional) .....	17
Imager(s) (Optional) .....	17
Software Bring-Up Checklist .....	18
Preparation .....	18
Bring-up Hardware Validation .....	18
U-Boot Port and Boot Validation .....	18
Kernel and Peripherals, Port and Validation .....	18
System Power and Clocks .....	19

# PLATFORM ADAPTATION AND BRING-UP GUIDE

This document is for software developers whose target is the NVIDIA® Jetson™ TX1 module. It describes how to port the NVIDIA® Tegra® Linux Driver Package and the U-Boot boot loader from NVIDIA® Jetson™ TX1 Developer Kit to other hardware platforms.

## PORTING LINUX FOR TEGRA (L4T) TO YOUR DESIGN

For all of the procedures below, the NVIDIA® Tegra® Linux Driver Package (also known as L4T: Linux for Tegra) release includes code for the Jetson TX1 Developer Kit (P2371-2180) that can serve as an example.

### Board Naming

To support your board in L4T, you must choose a simple lower-case, alpha-numeric name for your board, possibly including dashes (-) or underscores (\_) but containing no spaces, such as the following examples:

```
jetson-tx1  
p2371-2180  
beaver
```

The name you choose will appear in file names and path names in U-Boot and Linux kernel source code, user-visible device tree file names, and be exposed to the user via the U-Boot command prompt and various Linux kernel `/proc` files.

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

You must also choose 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 simply re-use and modify the existing NVIDIA® Jetson™ TX1 Developer Kit code without choosing and using your own board name. If you do not use your own board name it will not be obvious to Jetson TX1 users whether modified source code supports the original Jetson TX1 Developer Kit carrier board or your board.

## Placeholders in the Porting Instructions

The following sections refer to filenames and pathnames that contain these placeholders. Substitute an appropriate value for each placeholder when you enter the 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 chosen to represent your platform. For example, `p2597` is the name of the Jetson TX1 Developer Kit carrier board. Note that NVIDIA `<board>` names use lower case letters only.
- `<som>` is a System on a Module (SOM) board name, such as `2180`.
- `<version>` is a board version number, such as `a00`. Files for NVIDIA reference boards include a version number. Files for customer platforms need not include one.
- `<vendor>` is your organization's name, or the name of your board's vendor.
- `<root>` is the device that holds the platform's root file system. At present the only supported value is `emmc`.

## Pinmux Changes

If your board schematic differs from that for Jetson™ TX1 Developer Kit carrier board, you must change the pinmux configuration applied by the software.

To define your board's pinmux configuration, you must obtain `Jetson_TX1_customer_pinmux_release.xlsm` from NVIDIA and customize it for the configuration of your board using the following procedures.

### To customize the pinmux spreadsheet

1. Create a copy of the file with a name based on your board name, e.g. `<board>_pinmux.xlsm`.
2. Ensure that the new file is writable.

3. On a Windows PC, open the new file in Microsoft Excel.
4. If Microsoft Excel displays any warnings such as “PROTECTED VIEW” or “SECURITY WARNING,” click Enable Editing or Enable Content, so that you can save your changes to the new file.
5. Rename the **Jetson TX1 Configuration** tab based on the name of your board:
  - Right-click on the **Jetson TX1 Configuration** tab at the bottom of the window.
  - Click the **Rename** menu option.
  - Type your board name into the tab, then press ENTER.
6. Modify columns AE through AO of the spreadsheet as required by the pinmux configuration for your board, based on the schematic.

Once the spreadsheet reflects the configuration you want, export the configuration data in a format that U-Boot and the Linux kernel can use.

## Exporting Pinmux for U-Boot

U-Boot uses a header file to define the pinmux configuration. This header file may be generated using the `tegra-pinmux-scripts` tool.

### To customize `tegra-pinmux-scripts` for your board

1. Obtain `tegra-pinmux-scripts` by running the following commands on your Linux system:

```
$ git clone https://github.com/NVIDIA/tegra-pinmux-scripts.git
$ cd tegra-pinmux-scripts
```

2. In the `tegra-pinmux-scripts` directory, open `csv-to-board.py` in a text editor.
3. Locate the definition of the `supported_boards` data structure, at approximately line 50.
4. Add an entry for your board to the `supported_boards` data structure similar to the following example:

```
'<board>': {
    # <board>_pinmux.xlsm worksheet <board>
    'filename': 'csv/<board>.csv',
    'rsvd_0based': False,
},
```

5. Save the file and exit the editor.
6. Commit this change to your local Git repository:

```
$ git commit -a -m "Add support for <board>" -s
```

The `tegra-pinmux-scripts` read a CSV (Comma Separated Values) version of the pinmux spreadsheet as input.

### To save the spreadsheet in CSV format

1. In Microsoft Excel, click the File tab.
2. On the File tab, click Save As.
3. From Save as type, choose CSV (MS-DOC) (\*.csv).
4. Verify that the file name ends in `.csv`, but otherwise matches the file name in your changes to `csv-to-board.py`.
5. Click Save.
6. Copy the CSV file to the `csv/` directory of `tegra-pinmux-scripts` on your Linux system.

### To generate the U-Boot pinmux header file

1. Enter the following command in the `tegra-pinmux-scripts` directory to import the data into the `tegra-pinmux-script` internal format:

```
$ ./csv-to-board.py <board>
```

Optionally, use the `--csv <csv_file_name>` command line option to specify the CSV file to import. This allows you to copy the CSV file to an arbitrary location on your Linux system if you wish.

2. Enter the following command to generate the U-Boot pinmux header file:

```
$ ./board-to-uboot.py <board> > pinmux-config-<board>.h
```

Later, you will copy `pinmux-config-<board>.h` into the U-Boot source tree.

## Exporting Pinmux for the L4T Linux Kernel

The Linux kernel uses device tree files to define the pinmux configuration, which you can generate directly from the Excel spreadsheet.

### To generate device tree files for your pinmux configuration

1. In the spreadsheet, click Generate DT.
2. Answer “yes” to the prompt asking whether you wish to generate the DT files and provide the name of your board when prompted.

The device tree files are saved in the same location as the Excel spreadsheet. After the file is generated, make sure that the file name matches what you use in your kernel code.

Correct the file name if there is a mismatch. Later, you will copy the device tree files into the Linux kernel source tree.

## Porting U-Boot

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

1. Copy `include/configs/jetson-p2371-2180.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 P2371-2180"
```

3. Copy `arch/arm/dts/tegra210-p2371-2180.dts` to `arch/arm/dts/tegra210-<board>.dts`.
4. Edit the set of enabled devices and their parameters (e.g. GPIO and IRQ IDs) in `tegra210-<board>.dts` as appropriate for your board.

Nodes and properties might need to be added, removed, or edited.

**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 `tegra210-<board>.dtb` to `arch/arm/dts/Makefile`.
6. Copy `configs/p2371-2180_defconfig` to `configs/<board>_defconfig`.
7. Edit `<board>_defconfig` to refer to your board name.
8. Edit `arch/arm/mach-tegra/tegra210/Kconfig` to add target config and `Kconfig`.
9. Copy the `board/nvidia/p2371-2180/` directory to `board/<vendor>/<board>/`.
10. Edit all of the files in `board/<vendor>/<board>/` to refer to your board name rather than the P2371-2180. The files in this directory contain many instances of the P2371-2180 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.
13. Copy the pinmux header you generated (`pinmux-config-<board>.h`) to the `board/<vendor>/<board>/` directory.



## Porting the Linux Kernel

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

The configuration files are in `arch/arm64/boot/dts/`. Their names have the form `tegra210-jetson-cv-base-<board>-<som>.dts`, where `<board>` refers to one of the NVIDIA reference boards, such as `p2597`.

NVIDIA recommends that you use this file and the files included, which describes the device tree of reference board P2597:

```
arch/arm64/boot/dts/tegra210-jetson-cv-base-p2597-2180-a00.dts
```

This device tree file includes many `.dtsi` files for various types of hardware. To configure the kernel to work on your platform, make copies of the `.dts` file and the `.dtsi` files it references, and modify the copies to correspond to your platform's design.

The following procedure will guide you through this process.

1. Copy the `.dts` file you have chosen to this location:

```
arch/arm64/boot/dts/tegra210-<board>.dts
```

2. From the `arch/arm64/boot/dts/tegra210-platforms/` directory, copy each file whose name has the form:

```
tegra210-jetson-cv-<function>-<board>-<som>-<version>.dtsi
```

Copy each file to:

```
arch/arm64/boot/dts/tegra210-platforms/tegra210-<board>-<function>.dtsi
```

You may rename the copies if that makes your work easier.

3. Edit your copy of the `.dts` file to refer to your copies of the `.dtsi` files.
4. Edit the set of enabled devices and their parameters (e.g. GPIO and IRQ IDs) in each copied file as appropriate for your board.

You may need to add, remove, or edit `.dtsi` file 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 the Linux kernel rather than from U-Boot.

5. Replace the content of `tegra210-<board>-gpio.dtsi` and `tegra210-<board>-pinmux.dtsi` with the content you generated from the kernel pinmux files earlier.
6. Edit `arch/arm64/boot/dts/Makefile` to add an entry for your board, modeled after the existing Jetson™ TX1 entry.
7. Copy this file:

```
Linux_for_Tegra/bootloader/t210ref/p2371-2180/extlinux.conf.emmc
```

To this location:

```
Linux_for_Tegra/bootloader/t210ref/<board>-<som>/extlinux.conf.emmc
```

8. Edit the following statements in each copied file to refer to your board:

```
FDT /boot/tegra210-jetson-cv-base-p2597-2180-a00.dtb
```

9. Copy `Linux_for_Tegra/jetson-tx1.conf` to `Linux_for_Tegra/<board>.conf`.
10. Edit `SYSBOOTFILE` and `DTB_FILE` in `<board>.conf` to refer to your board.

Following are some examples of modifications that you may have to make.

## Regulator

VDDIO of SDMMC1 comes from PMU LDO2. That section looks like this:

```
regulators {
    ldo2 {
        regulator-name = "vddio-sdmmc1";
    };
};
```

If there is any change for this power tree, this item should be changed.

## Pad Power Detection

Pad power on the T210 defaults to 3.3 V. Unlike some other Tegra models, the T210 does not have auto power detect cells. I/O pads that are powered at 1.8 V must be set manually in DTS to 1.8 V.

```
gpio {
    nvidia,io-pad-init-voltage = <IO_PAD_VOLTAGE_1_8V>;
};
```

## GPIO

There are many GPIO configurations in different hardware modules. To change the GPIO setting, check the related device tree file.

For example, VDD of SDMMC is controlled by a GPIO pin (GPIO\_PZ3). That is a power tree module, so the definition of this part is in:

```
tegra210-platforms/tegra210-jetson-cv-power-tree-p2597-2180-a00.dtsi
```

And looks like this:

```
en_vdd_sd: regulator@4 {
    gpio = <&gpio TEGRA_GPIO(Z, 3) 0>;
};
```

You can change this setting according to platform circuit.

## Interrupt

For modules that make interrupt requests, the interrupt requests can also be declared in the .dts file.

For example, this file:

```
tegra210-platforms/tegra210-comms-p2530-0930.dtsi
```

Describes a WIFI interrupt like this:

```
bcmhdhd_wlan {
    compatible = "android,bcmhdhd_wlan";
    interrupt-parent = <&gpio>;
    interrupts = <TEGRA_GPIO(H, 2) 0x14>;
    wlan-pwr-gpio = <&gpio TEGRA_GPIO(H, 0) 0>;
    status = "okay";
};
```

This specifies GPIO\_PH02 (same pin as WIFI\_WAKE\_AP) as the interrupt request pin from the WIFI module.

## Key

The key is defined in:

```
tegra210-platforms/tegra210-keys-p2530-0930.dtsi
```

The power key is defined as:

```
power {
    label = "Power";
    gpios = <&gpio TEGRA_GPIO(X, 5) GPIO_ACTIVE_LOW>;
    linux,code = <KEY_POWER>;
    gpio-key,wakeup;
};
```

**Note:** This example is meant only to show how to define a key in a .dts file. 'Power key' is a special key, and *cannot* be changed.

For the detailed information about .dts files, refer to the documentation at [Documentation/devicetree/bindings](#) in the NVIDIA released Linux kernel source package.

## Other Considerations When Porting

This section describes other considerations and recommendations to consider when porting.

### To flash the build image

- When flashing the build image, use your specific board name:

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

### To flash with BOARDID if the design does not use EEPROM

BOARDID is either passed using an XML file during flashing or is read from EEPROM. The flashing software uses the BOARDID from the XML file if provided; otherwise it uses the EEPROM value. The file `board_config_p2597-devkit.xml`, shown below, illustrates the XML file format.

```
<?xml version="1.0"?>
<!-- Nvidia Tegra board info configuration file -->
<board_configs>
    <board type="proc" id="2180" sku="1000" fab="0" />
    <board type="display" id="0000" sku="0000"/>
    <board type="pmu" id="2180" sku="0000" />
</board_configs>
```

This flashing config file `p2371-2180-devkit.conf` passes the name of the XML file as an option:

```
BCFFILE="bootloader/${target_board}/cfg/board_config_p2597-devkit.xml";
```

The file contains the processor module ID (`type="proc"`), display board ID (`type="display"`), and power management unit ID (`type="pmu"`). Since the processor and PMU are on the same module in the development kit, they have the same ID.

If you add new values for the `board` tag's `id` property, you must add them to the list of valid values in `nvtboot`.

## To change the UART port to UARTA

1. In `Linux_for_Tegra/<board>.conf`, modify the ODMDATA assignment:

```
ODMDATA=0x60084000;
```

2. In the U-Boot boot loader, locate the following lines in `/include/configs/jetson-tx1.h`:

```
#define CONFIG_TEGRA_ENABLE_UARTD
#define CONFIG_SYS_NS16550_COM1          NV_PA_APB_UARTD_BASE
```

3. Modify those lines to specify UARTA:

```
#define CONFIG_TEGRA_ENABLE_UARTA
#define CONFIG_SYS_NS16550_COM1          NV_PA_APB_UARTA_BASE
```

4. In the kernel, modify the `debug_uartport` assignment:

```
debug_uartport=1sport,0
```

## Hardware Bring-Up Checklist

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

### Before Power-On

Make sure that the Jetson TX1 is connected to the BTB connector correctly and securely.	<input type="checkbox"/>
---	--------------------------

Verify that power supplies are not shorted to ground or to other power supplies.	<input type="checkbox"/>
--	--------------------------

## Initial Power-On

Verify that VDD_IN from carrier board is in the 6 V to 19 V range.	<input type="checkbox"/>
Verify that CARRIER_WR_ON goes to HIGH when power is turned on.	<input type="checkbox"/>
Verify that system can enter force recovery.	<input type="checkbox"/>

## Initial Software Flashing

Verify that system can be flashed with TegraFlash.	<input type="checkbox"/>
Verify that TegraBoot and U-boot run to completion by checking log output.	<input type="checkbox"/>
Verify that OS runs to desktop.	<input type="checkbox"/>
Verify that any UARTs intended for debugging are enabled and functional.	<input type="checkbox"/>

## Power

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

## Power Optimization

Capture CPU PWR Request entering and exiting Suspend (LP1) and Deep Sleep (LP0). Ensure that CPU PWR Request and associated power rail sequence meets Tegra Data Sheet requirements.	<input type="checkbox"/>
Verify that all rails which must be OFF in Deep Sleep (LP0) are OFF.	<input type="checkbox"/>
Verify that all rails which must be ON in Deep Sleep (LP0) are ON.	<input type="checkbox"/>
Verify that required rails are back and at correct voltage under hardware control exiting Deep Sleep (LP0).	<input type="checkbox"/>

## USB 2.0 PHY

Verify that USB0 supports USB Recovery (device mode).	<input type="checkbox"/>
Verify that USB0 device mode works with intended peripheral types, if supported.	<input type="checkbox"/>
Verify USB0, USB1 and or USB2 Host mode, if implemented.	<input type="checkbox"/>
Verify USB0 Device/Host detection, if supported.	<input type="checkbox"/>
Verify that USB PHYs go to lowest power mode when not used or when the system is in	<input type="checkbox"/>

low power mode.	
Verify that AVDD_USB and AVDD_PLL_UTMIP are off during Deep Sleep (LP0).	<input type="checkbox"/>
Capture USB0_D+/D- signals at both ends of link (connector and test points near Tegra).	<input type="checkbox"/>
Capture USB2_D+/D- signals at both ends of link (connector and test points near Tegra).	<input type="checkbox"/>
Using USB-IF procedures, verify that signals meet requirements (correct eye height/width, etc.).	<input type="checkbox"/>
If USB signals do not meet requirements, use the <i>Tegra USB Tuning Guide</i> to adjust settings until requirements are met.	<input type="checkbox"/>

## USB 3.0

Verify USB 3.0 Host mode.	<input type="checkbox"/>
Verify USB 3.0 Device mode, if enabled.	<input type="checkbox"/>
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.	<input type="checkbox"/>

## HDMI

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

## Audio

Verify reads and writes on I2C interface used for Audio Codec.	<input type="checkbox"/>
Verify that playback works properly on speakers, headphones, and headset.	<input type="checkbox"/>
Verify that capture works properly: Sound is recorded from microphone/headset if supported.	<input type="checkbox"/>
Verify that tones, voice, etc. can be heard from speakers or headphones/headset.	<input type="checkbox"/>
Verify that Audio Codec goes to lowest power mode when not in use or system enters low power mode.	<input type="checkbox"/>
Capture signals at receiver end of link, if accessible, for each I2S I/FT used.	<input type="checkbox"/>
Verify that signal quality is acceptable. Look for excessive over/undershoot and glitches on signal edges.	<input type="checkbox"/>

## UART

Verify that Tegra TX/RX/CTS/RTS connects to device RX/TX/RTS/CTS for each UART used.	<input type="checkbox"/>
Verify that signal quality is acceptable. Look for excessive over/undershoot and glitches on signal edges.	<input type="checkbox"/>

## SD Card (SDMMC1)

Verify proper connectivity by setting Tegra pins to GPIOs, if necessary, to debug.	<input type="checkbox"/>
Verify that basic SD commands operate properly.	<input type="checkbox"/>
Verify reads and writes for a variety of SD Cards.	<input type="checkbox"/>
Verify that SD Card insertion detection works and wakes system, if supported.	<input type="checkbox"/>
Verify that SD Card Write Protect works, if implemented.	<input type="checkbox"/>
Verify that SD Card goes to low power mode or rails are powered off when not used or in low power system state.	<input type="checkbox"/>
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.	<input type="checkbox"/>

## Sensors I2C: General

Verify that addresses of all I2C devices appear correctly, and no unknown ghost devices appear.	<input type="checkbox"/>
Verify that signal quality is acceptable, including rise times of signals, when probed at BTB connector and devices.	<input type="checkbox"/>

## 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).	<input type="checkbox"/>
Verify that interrupts are generated properly.	<input type="checkbox"/>
Verify functionality of Touch Screen.	<input type="checkbox"/>
Verify that Touch Screen Controller goes to lowest power mode when not used, or system is in low power state.	<input type="checkbox"/>

## PEX (Optional)

Verify proper connectivity by checking lanes.	<input type="checkbox"/>
Verify that any implemented PEX interfaces transition to the lowest power state in Deep Sleep (LP0) and Suspend (LP1).	<input type="checkbox"/>
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.	<input type="checkbox"/>



## SATA (Optional)

Verify proper connectivity by checking diff lines.	<input type="checkbox"/>
Verify that any implemented SATA interfaces transition to the lowest power state in Deep Sleep (LP0) and Suspend (LP1).	<input type="checkbox"/>
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.	<input type="checkbox"/>

## Embedded Display(s) (Optional)

Verify that I2C or other control interface is able to perform writes/reads to display.	<input type="checkbox"/>
Verify that each embedded display shows correct colors.	<input type="checkbox"/>
Verify that each embedded display's backlight is enabled when in normal display mode.	<input type="checkbox"/>
Verify that each embedded display's backlight brightness can be adjusted properly.	<input type="checkbox"/>
Verify that each embedded display's backlight is disabled when in a low power mode.	<input type="checkbox"/>
Verify that each embedded display (and any display bridge) transitions to the lowest power state in Deep Sleep (LP0) and Suspend (LP1).	<input type="checkbox"/>
Verify that power-on/off sequencing of rails associated with each display meets manufacturer's requirements.	<input type="checkbox"/>
Verify DSI, LVDS or eDP timing (see <i>Tegra DC and DSI Debugging Guide</i> for details on how and what to verify).	<input type="checkbox"/>
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.	<input type="checkbox"/>

## Imager(s) (Optional)

Verify that I2C interface writes/reads work to all cameras.	<input type="checkbox"/>
Verify that preview displays properly for all cameras.	<input type="checkbox"/>
Verify that still capture works on all cameras.	<input type="checkbox"/>
Verify that video capture works on all cameras.	<input type="checkbox"/>
Verify that all flashes operate properly.	<input type="checkbox"/>
Verify that any available autofocus mechanism functions properly.	<input type="checkbox"/>
Verify that privacy LED operates properly, if implemented.	<input type="checkbox"/>
Verify that cameras and related circuitry enter lowest power mode when not used or system is in a low power mode.	<input type="checkbox"/>
Verify that power-on/off sequencing of rails associated with imager module meets manufacturer's requirements.	<input type="checkbox"/>
Probe MCLK output at recommended test points, and verify that signal quality is acceptable. Look for excessive over/undershoot and glitches on signal edges.	<input type="checkbox"/>

## Software Bring-Up Checklist

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

### Preparation

Verify Board BCT.	<input type="checkbox"/>
Verify operation eMMC with the NVIDIA Diagnostic Tool.	<input type="checkbox"/>
Obtain board schematics and component data sheets.	<input type="checkbox"/>
Verify power tree.	<input type="checkbox"/>
Review board pinmux.	<input type="checkbox"/>

### Bring-up Hardware Validation

Power and Reset Sequence, Power Rail Check	<input type="checkbox"/>
Recovery Mode	<input type="checkbox"/>
NvTest (Tegra MODS) DDR, eMMC, CPU	<input type="checkbox"/>
JTAG connection check	<input type="checkbox"/>

### U-Boot Port and Boot Validation

TegraFlash	<input type="checkbox"/>
UART output	<input type="checkbox"/>
KBD connection	<input type="checkbox"/>
Board config/PMIC regulator config/Pinmux/Review device tree	<input type="checkbox"/>
Verify FS support/Config boot scripts (bootcmd)	<input type="checkbox"/>
Boot to U-boot	<input type="checkbox"/>
Boot to kernel	<input type="checkbox"/>
Boot to kernel command line or custom desktop	<input type="checkbox"/>

### Kernel and Peripherals, Port and Validation

Device tree review, Pinmux, GPIO, Wake pins	<input type="checkbox"/>
PMU and regulator drivers	<input type="checkbox"/>
Display/HDMI	<input type="checkbox"/>
Audio codec	<input type="checkbox"/>
Microphone and speaker	<input type="checkbox"/>
USB	<input type="checkbox"/>
SD card	<input type="checkbox"/>

Thermal Sensor	<input type="checkbox"/>
EMC DFS table	<input type="checkbox"/>
Ethernet	<input type="checkbox"/>
SATA	<input type="checkbox"/>
PCIe	<input type="checkbox"/>

## System Power and Clocks

CPU/CORE/GPU DVFS	<input type="checkbox"/>
EMC DFS table	<input type="checkbox"/>
CPU/CORE EDP	<input type="checkbox"/>
GPU EDP	<input type="checkbox"/>
System EDP (Contain Current monitor & Voltage comparator)	<input type="checkbox"/>
Power Off	<input type="checkbox"/>
LP0 (optional)	<input type="checkbox"/>
CPU power down (LP2)	<input type="checkbox"/>
BCT, Full-speed	<input type="checkbox"/>

## Notice

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, EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE MATERIALS, AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OR CONDITION OF TITLE, MERCHANTABILITY, SATISFACTORY QUALITY, FITNESS FOR A PARTICULAR PURPOSE AND ON-INFRINGEMENT, ARE HEREBY EXCLUDED TO THE MAXIMUM EXTENT PERMITTED BY LAW.

Information furnished is believed to be accurate and reliable. However, NVIDIA Corporation assumes no responsibility for the consequences of use of such information or for any infringement of patents or other rights of third parties that may result from its use. No license is granted by implication or otherwise under any patent or patent rights of NVIDIA Corporation. Specifications mentioned in this publication are subject to change without notice. This publication supersedes and replaces all information previously supplied. NVIDIA Corporation products are not authorized for use as critical components in life support devices or systems without express written approval of NVIDIA Corporation.

## Trademarks

NVIDIA, the NVIDIA logo, Tegra, and Jetson are trademarks or registered trademarks of NVIDIA Corporation in the United States and other countries. Other company and product names may be trademarks of the respective companies with which they are associated.

## Copyright

© 2016 NVIDIA Corporation. All rights reserved.