# Jetson Nano Developer Kit 40-Pin Expansion Header Configuration

Application Note

DA_09565-001 | August 2019

# Document History

Doc_Number

| Version | Date | Authors | Description of Change |
|---|---|---|---|
| 1.0 | July 9, 2019 | jsachs, jonathanh, twarren | Initial release |
| 1.1 | July 24, 2019 | jsachs, jonathanh, twarren | Minor corrections |
| 1.2 | August 7, 2019 | jsachs, jonathanh, twarren | Minor corrections |

# Table of Contents

# Customizing the Jetson Nano 40-pin Expansion Header

The Jetson Nano Developer Kit carrier board includes a 40-pin expansion header. By default, all interface signal pins are configured as GPIO inputs, except pins 3 and 5 and pins 27 and 28, which are I2C SDA and SCL, and pins 8 and 10, which are UART TX and RX.

This application note describes how to alter the function of pins on the 40-pin header by using the Jetson Nano Developer Kit pinmux spreadsheet. Note that the pinmux actually configures the SoC on the Jetson module, which ultimately routes the SoC signals to the carrier board's 40-pin header.

If you want to configure other SoC pins, see the full Jetson Nano module pinmux spreadsheet and the NVIDIA L4T Development Guide for complete documentation.

## Prerequisites

- A Jetson Nano Developer Kit.
- A computer running Linux (a **Linux host**) which has the GCC toolchain installed that is recommended for building L4T. For more details, see the topic "The L4T Toolchain" section in the *L4T Development Guide*.
- A computer running Microsoft Windows (a **Windows host**) with Microsoft Excel installed.

## Download and Customize the Pinmux Spreadsheet

1. On the Windows host, download the Jetson Nano Developer Kit Pinmux spreadsheet from the Jetson Download Center.
2. Open the file in Microsoft Excel and ensure that:
    - The spreadsheet file is writable.
    - The Excel option "Enable Editing" or "Enable Content" is selected. You may need to set one of these options if Excel displays warnings such as "PROTECTED VIEW" or "SECURITY WARNING."
    - The spreadsheet macros are enabled. The spreadsheet needs them to generate device tree source files.
3. Modify columns AR ("Customer Usage") and AS ("Pin Direction") to change the function of individual pins on the developer kit's 40-pin header.
4. Check the following columns through BA to see if any other values must be adjusted.

You need not modify column BB ("IO Block Voltage," concerning SoC connector voltages), since all signals on the carrier board's 40-pin expansion header use 3.3V.

5. Click the Generate DT file button to export your pinmux configuration. When prompted, enter the board name `jetson-nano-sd`. The spreadsheet creates two device tree source files, which you will use in the section "Update the CBoot Pinmux" below.
6. Save the spreadsheet in CSV (comma-delimited) format. You will use this CSV file in To update the CBoot pinmux.

   To save the spreadsheet in CSV format:

   1. Under the File menu in Excel, select Save As.
   2. Enter the filename `jetson-nano-sd.csv`.
   3. Select CSV UTF-8 (comma-delimited) (*.csv) from the menu of filetypes.
   4. Click Save.

## Download the L4T Driver Package and Source Files

On your Linux host, download and extract the L4T Jetson Driver Package.

Go to the directory `Linux_for_Tegra` and run the script `source_sync.sh` to download the various source trees. When prompted, enter the correct tag for the L4T version you are using. The release tag name can be found in the L4T Release Notes. For example, for L4T release 32.1, enter the tag `tegra-l4t-r32.1`.

```
$ cd Linux_for_Tegra/
$ ./source_sync.sh
```

## Update the U-Boot Pinmux

The pinmux configuration programmed by the U-Boot bootloader is stored in one of the U-Boot header files. Therefore, you must update the U-Boot header, rebuild U-Boot, and reflash. Note that the header file is generated using the `tegra-pinmux-scripts` tool to parse the CSV version of the pinmux spreadsheet. To generate the header file and update the U-Boot bootloader:

1. On your Linux host, obtain a copy of `tegra-pinmux-scripts` by cloning the following Git tree:

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

2. On your Linux host, place a copy the CSV file in the `tegra-pinmux-scripts/csv` directory.

   ```
   $ cd tegra-pinmux-scripts
   $ mkdir csv
   $ cp <path-to-csv>/jetson-nano-sd.csv csv/p3450-porg.csv
   ```

Where `<path-to-csv>` is the pathname of the directory that contains the CSV file you created in To download and customize the pinmux spreadsheet. This assumes that the CSV file's directory is accessible from the Linux host. If it is not, copy the file by some other means.

3. Import the pinmux CSV file into the `tegra-pinmux-script` internal format:

```
$ ./csv-to-board.py p3450-porg
```

If the following error occurs, it is most likely because the filetype was set to "CSV..." instead of "CSV UTF-8... " when you exported the spreadsheet to CSV.

```
UnicodeDecodeError: 'utf-8' codec can't decode byte 0x96 in position
5617: invalid start byte
```

Make sure that the spreadsheet is exported using the correct CSV format.

4. Generate the U-Boot pinmux header file:

```
$ ./board-to-uboot.py p3450-porg > pinmux-config-p3450-porg.h
```

5. Rebuild the U-Boot bootloader.
   1. Copy the U-Boot header file generated in the previous step to this directory:

```
$ cd Linux_for_Tegra/sources/u-boot/
$ cp <path-to-pinmux-scripts>/tegra-pinmux-scripts/pinmux-config-
p3450-porg.h board/nvidia/p3450-porg/
```

   2. Set the build environment:

```
$ export CROSS_COMPILE=<toolchain_install_path>/bin/aarch64-linux-gnu
```

   3. Build U-Boot by running the commands:

```
$ make distclean
$ make p3450-porg_defconfig
$ make
```

   4. Copy the new U-Boot binary to the L4T tree, where it will be found and used when flashing:

```
$ cp u-boot.bin ../../bootloader/t210ref/p3450-porg/
```

## Update the CBoot Pinmux

The CBoot bootloader uses device tree files generated by the Excel spreadsheet to configure the pinmux. The pinmux settings in device tree files are only applied by CBoot, and not re-applied by the Linux kernel. To use the updated device tree files you must rebuild the device tree image for Jetson Nano. To update the device tree image:

1. Determine the Jetson Nano device tree version. You can determine the version from the device tree source file name, which can be found by executing this command on Jetson Nano:

   ```
   $ cat /proc/device-tree/nvidia,dtsfilename
   ```

   If the file name is `tegra210-p3448-0000-p3449-0000-a02.dts`, the version is a02. If it is `tegra210-p3448-0000-p3449-0000-b00.dts`, the version is b00.

2. For L4T release 32.1 only, you must execute the following commands to update the device tree source before applying the pinmux changes:

   ```
   $ cd Linux_for_Tegra/sources/hardware/nvidia/platform/t210/porg/kernel-dts/porg-platforms/
   $ git fetch --all
   $ git cherry-pick 3082c5efb5c1dabaaae27df666715760d2634fa7
   $ git cherry-pick d31ba3a0dd913760600d7497e8f7317300821bc0
   ```

   If you don't make these changes for L4T release 32.1, the device tree compiler issues this message when it builds the device tree image:

   ```
   ERROR (phandle_references): Reference to non-existent node or label "suspend_gpio"
   ```

3. Copy the newly generated device tree files from your Windows host to your Linux host and place them in the following locations under the L4T Driver Package.

   ```
   $ cd Linux_for_Tegra/sources/hardware/nvidia/platform/t210/porg/kernel-dts/porg-platforms/
   $ cp <path-to-new-dt-files>/tegra210-jetson-nano-sd-pinmux.dtsi tegra210-porg-pinmux-p3448-0000-<nano-dt-version>.dtsi
   $ cp <path-to-new-dt-files>/tegra210-jetson-nano-sd-gpio-default.dtsi tegra210-porg-gpio-p3448-0000-<nano-dt-version>.dtsi
   ```

   Where `<nano-dt-version>` is the version that you determined in the previous step.

If you don't make this change to the `tegra210-porg-gpio-p3448-0000-<nano-dt-version>.dtsi` file, the following error will occur when you build the device tree image:

```
ERROR (phandle_references): Reference to non-existent node or label "suspend_gpio"
```

4. Set the build environment:

```
$ export CROSS_COMPILE=<toolchain_install_path>/bin/aarch64-linux-gnu
```

5. Re-build the device tree image:

```
$ cd Linux_for_Tegra/sources/kernel/kernel-4.9/
$ make ARCH=arm64 tegra_defconfig
$ make ARCH=arm64 dtbs
```

6. Copy the updated device tree image to the L4T tree:

```
$ cp arch/arm64/boot/dts/tegra210-p3448-0000-p3449-0000-<nano-dt-version>.dtb ../../../kernel/dtb/
```

## Flash Jetson Nano

Use your Linux host to flash the updated U-Boot bootloader and device tree image to Jetson Nano. Remember that Jetson Nano must first be placed in Force Recovery mode.

```
$ cd Linux_for_Tegra/
$ sudo ./flash.sh jetson-nano-qspi-sd mmcblk0p1
```

## Notice



## Trademarks



## Copyright



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