Tegra Linux Driver Package L4T.ER1

RN_05071-001_12 | June 24, 2011

Release Notes

www.nvidia.com

TABLE OF CONTENTS

TEGRA LINUX DRIVER PACKAGE RELEASE NOTES

These release notes provide information about package contents, supported features, and known issues for this release of the NVIDIA® Tegra Linux Driver Package.

1.0 About this Release

Tegra Linux Driver Package supports development of platforms running:

- NVIDIA® Tegra™ 200 series computer-on-a-chip
- The chromeos-2.6.38 branch of the Linux kernel available here:

  http://git.chromium.org/gitweb/?p=chromiumos/third_party/kernel.git;a=shortlog;h=refs/heads/chromeos-2.6.38

Note: This alpha release of Tegra Linux Driver Package (L4T.ER1) is a release for Tegra 250 "Harmony" only. The information in these release notes is preliminary.

1.1 Supported "Alpha" features

The following list shows implemented features that are supported in this Alpha pre-release.

- Kernel version 2.6.38
- Media APIs:
  - OpenGL ES 2.0
  - OpenGL ES 1.1
  - Open GL ES path extensions

- EGL 1.3 with EGLImage

- OpenMAX IL 1.1
- X Resize, Rotate and Reflect Extension (RandR) 1.3

1.2 Implementation Notes

This section provides additional implementation and support information specific to this release of Tegra Linux Driver Package.

1.2.1 Logging in

When using the provided sample filesystem, log in as user ”ubuntu”. Both the password and the sudo password are “ubuntu”. For more information, see Figure 3.4.1, “About the Root File System,” on page 6.

1.2.2 64-bit Ubuntu requires 32-bit runtime support

If you are running a 64-bit Ubuntu installation on the host PC, this release requires that you have 32-bit runtime support installed as well. The simple steps for installing 32-bit support on an Ubuntu host use the apt-get command.

To install 32-bit runtime support

- Execute the following commands:

```
$ sudo apt-get update
$ sudo apt-get install ia32-libs
```

Note: If `ia32-libs` is not installed, flashing the boot loader with the kernel (and, possibly, other steps in the process) will fail.

1.2.3 Display mode and resolution configuration with the xandr application

You can use the X Resize, Rotate and Reflect Extension (RandR) extension to manipulate and configure the attached displays (both the internal panel and any externally connected HDMI panel). The `xrandr`(1) utility is the most common way to do this.

A tutorial on `xrandr` can be found on the following website:

`http://www.thinkwiki.org/wiki/Xorg_RandR_1.2`

1.2.4 ALSA configuration to allow recording

This implementation note documents how to configure the Advanced Linux Sound Architecture (ALSA) to enable recording. It pertains to both the Harmony and Ventana boards.

To configure ALSA for recording

1. Execute the following command:

   ```
   $ alsamixer
   ```

2. Adjust these settings as follows:
   - Left Capture Mux: Left
   - Right Capture Mux: Left
   - Left Input PGA: Unmute, turn volume up (i.e., to the maximum)
   - Left Inverting Input Mux: IN1L

To verify recording functionality

▶ Execute the following command:

```
$ arecord -f dat -d 60 <path_to_testfile>.wav
```

2.0 Known Issues

The following list describes known issues with this release:

▶ The power button must be used to wake from the Suspend (LP1) power state. This is because Suspend shuts off USB power, so Resume fails to wake up the target board from the USB keyboard/mouse and crashes the system. The workaround is to wake up the system using the power button.

▶ CRT output support is not present in the kernel for Harmony. Only LVDS and HDMI are available.

▶ HDMI on Harmony will not display until you log in over a serial or network connection and start X.

▶ Wi-Fi is not present on Harmony.

▶ There is no NAND or NFS boot functionality in this release.

▶ Deep Sleep (LP0) does not work.

3.0 Documentation

The following sections contain information to help you get started using this pre-release of Tegra Linux Driver Package. They cover the following topics:

3.1 Requirements

- Host PC running Linux. Ubuntu 9.04 is used in examples in this document, but other distributions should also work.
- Tegra Linux Driver Package does contain a kernel image (zImage), and the ability to download and build from source is also available in the package. For more information, see 3.6 "Rebuilding the Kernel" (page 9) in this guide.
- Flashing on Harmony requires the Fastboot utility, which is included in this release.

For more information, see the `3.3.3 "Additional Requirements" (page 5)` subsection below.

3.2 Boot Options

It is currently possible to boot Tegra Linux Driver Package Tegra 200 Series developer board ("Harmony") with a root file system from:

- USB disk
- eMMC
- NAND

3.3 Setting Up Your Environment

The following subsections contain information to help you get started using this pre-release of Tegra Linux Driver Package. They cover the following topics:

3.3.1 Extracting Tegra Linux Driver Package

Note: The procedures in this document assume you extract the release package in ~/.

To extract Tegra Linux Driver Package

1. Extract the package manually by executing the following command:

   ```
   $ tar -xvzf tegra-linux-12.alpha.1.0.tar.gz
   ```

2. In your environment for the Tegra 200 Series developer board ("Harmony") set:

   ```
   $ export TARGET_BOARD=harmony
   ```

3.3.2 Setting Up Your Board

Tegra Linux Driver Package requires either a Tegra 200 Series developer board ("Harmony") with a NAND module as well as a host PC running Linux.

To set up your developer board

1. Connect the power cord to the J1 slot.
2. Connect a USB cable from the host PC to the USB1 connector on the board.
3. Connect an RS-232 null-modem: cable between the host PC and the UART1 connector.

The Tegra 200 Series developer board ("Harmony") UART can be found on the satellite board.

4. Open a terminal on the host PC and set it up at:
   - 115200 baud
   - 8-bit
   - Parity none
   - 1 stop bit

3.3.3 Additional Requirements

You need a USB disk formatted to EXT3. (It can be a memory card with a USB adapter, or a USB key.) You also need a micro-USB male-to-USB Std A female cable (such as Hirose Electric Co Ltd ZX40-A-5S-75-STDAJ) to plug in to the board's USB3 connector.

3.4 Setting Up Your File System

This section describes the steps for setting up your file system. You must set up the root file system and copy the file system to USB disk. The following subsections cover the following topics:

3.4.1 About the Root File System

To replicate the sample file system, the `ttyS0.conf` and `nv.conf` files have been added. Both are accessible from the provided sample file system and are placed in the user-generated file system, should you decide to replicate the rootfs.

The provided sample root file system was created with Rootstock 0.1.99.4 using this command:

```
$ sudo rootstock --fqdn tegra-ubuntu --login ubuntu --password
ubuntu --imagesize 1G -d natty --seed ubuntu-minimal,xserver-xorg-
core,xinit,xterm,alsa-utils,wireless-tools,wpasupplicant,x11-
xserver-utils
```

This creates a file system with the hostname "tegra-ubuntu", the username "ubuntu" and the password "ubuntu".

The following packages are installed by default:

- ubuntu-minimal
- xserver-xorg-core
- x11-xserver-utils
- xinit
- xterm
- alsa-utils
- wireless-tools
- wpasupplicant

Note: You may want to install additional packages as described in the next procedure. For example, ssh will be useful if you need remote login.

To install more packages

Note: Prerequisite: attach an ethernet cable to the device through either the ethernet port (if available) or through the USB ethernet adaptor.

1. Once you have booted the target, turn on networking:

   ```
   $ sudo dhclient
   ```

2. Install packages using apt-get. For example, to install wget execute this command:

   ```
   $ sudo apt-get install wget
   ```

3.4.2 Setting Up the Root File System

The second step in booting the target board is to configure the root filesystem. Follow the steps in this procedure to set up the rootfs.

Note: The instructions below use the sample filesystem that is provided by NVIDIA as the base. If you would like to use your own, set the `LDK_ROOTFS_DIR` environment variable to point to where your rootfs is located and skip Steps 1 and 2.

To set up the rootfs

Note: As a prerequisite, you will need root/sudo access because the filesystems contain device nodes and certain permissions. The sample root filesystems are Ubuntu filesystems as regards both user and password, so you cannot log in directly as root, but rather need to use sudo.

1. Download the sample rootfs that NVIDIA provided.
2. Extract that rootfs to the `rootfs` directory included in this package.
3. Run the `apply_tegra_binaries.sh` script to copy the NVIDIA user space libraries into the target filesystem:

   ```
   $ ./apply_tegra_binaries.sh
   ```

   If you are using a different rootfs, or if you already have configured your rootfs, you can apply the NVIDIA user space libraries by setting the `LDK_ROOTFS_DIR` environment variable to point to your rootfs. Then run the script, as shown above, to copy the binaries into your target filesystem.

   To install a different X driver Application Binary Interface (ABI) version, you can use the `apply_tegra_X_abi.sh` script. Execute the script as follows:

```
$ ./apply_tegra_X_abi.sh #
```

where # is the ABI version you'd like to install, either 5, 6, 7, 8, or 10.

To determine the X driver ABI of the X server used in the root filesystem, start X once on the Tegra device. The resulting file `/var/log/Xorg.0.log` will contain something like the following:

```
(II) Module ABI versions:
X.Org ANSI C Emulation: 0.4
X.Org Video Driver: 8.0
```

The X.Org Video Driver line reports the ABI version. The sample Ubuntu 11.04-based root filesystem uses X driver ABI 10.

4. Load the target filesystem that you have generated onto the first partition of a device (either a USB stick, an SD card, or a USB hard drive) and attach that device to the target board. Power on the target board and the image should load.
5. You should have access to the board over a serial port. Open a terminal on the host PC and set it up at:
   - 115200 baud
   - 8-bit
   - Parity none
   - 1 stop bit

To copy the file system to USB disk

1. Plug your USB disk into the host PC.
2. Format it with an Ext3 file system and mount it, if necessary. We assume it is mounted to <mntpoint>.
3. Copy the file system.

   If `LDK_ROOTFS_DIR` is set, execute this command:

   ```
   $ sudo cp -r -p $LDK_ROOTFS_DIR/* <mntpoint>
   ```

   If it is not set, copy the `rootfs` directory that is included in the release by executing the following command:

   ```
   $ sudo cp -r -p <<LDK-directory>>/rootfs/* <mntpoint>
   ```

Once you have flashed your board, you can then unmount the disk and plug it to the board. For more information about:

- Flashing, see the 3.5 "Flashing the Boot Loader and Kernel" (page 9) section in this guide.

- Plugging in the board, see the hardware documentation for your developer board.

3.5 Flashing the Boot Loader and Kernel

There are two steps that must be taken to boot the target board (Harmony or Ventana). The first step is to flash the board with the boot loader and kernel.

The procedures in this section assume the following directories are present:

- `bootloader` - boot loader plus flashing tools (NvFlash, CFG, BCTs, etc.)
- `kernel` - has a kernel zImage, and scripts to sync/build the kernel
- `rootfs` - will house the root filesystem that you download
- `nv_tegra` - NVIDIA Tegra user space binaries

To flash the boot loader and kernel

1. You must first put the target board into reset/recovery mode. Do so by first powering on the board and then holding the recovery button and pressing the reset button.
2. Now run the `flash.sh` script that is in the top level directory of this release. The script must be supplied with the device name that it will have in the root filesystem.

   If the root filesystem will be on a USB disk, execute the script as follows:

   ```
   $ ./flash.sh sda1
   ```

   Otherwise, if the root filesystem will be on an SD card, execute the script as follows:

   ```
   $ ./flash.sh mmcblk0p1
   ```

   The boot loader and kernel will load.

3.6 Rebuilding the Kernel

You can manually rebuild the kernel used for this package. Internet access is a prerequisite to doing so. All the related scripts are located in the `kernel` directory.

To rebuild the Kernel

1. Get the kernel source by running the `kernel_sync.sh` script:

   ```
   $ cd ./kernel/
   $ ./kernel_sync.sh
   ```

2. Install the tool chain by running the `install_3rdparty.sh` script:

   ```
   $ ./install_3rdparty.sh
   ```

3 Compile the kernel by running the `kernel_build.sh` script:

```
$ ./kernel_build.sh
```

3.6.1 Configuration requirements for the chromeos-2.6.38 kernel

To replicate the kernel build you must set the following kernel configuration options:

- Enable the `CONFIG_FRAMEBUFFER_CONSOLE` variable. You can find this variable at:

  ```
  <kernel>/arch/arm/configs/tegra_defconfig
  ```
- Enable the `CONFIG_USB_NET_SMSC95XX` variable. You can find this variable at:

  ```
  <kernel>/arch/arm/configs/tegra_defconfig
  ```
- Replace all modules with drivers included in the kernel.