NVIDIA Tegra
Android Platform Support Pack
Getting Started Guide

Version 5421622

Contents

Introduction

The Android Tegra OS Support Pack for the NVIDIA Tegra 250 devkit is designed to provide all Tegra-specific components required to support development and use of a Tegra 250 devkit with Android. It contains the software required to flash and boot the devkit into Android.

This Support Pack is ***not*** a full Software Development Kit (SDK). It does not contain samples or support libraries. It contains only the NVIDIA-specific tools to flash and boot Android on the Tegra 250 devkit. To develop for Android, Google provide an SDK (http://developer.android.com/sdk/index.html) and a Native Development Kit (NDK) (http://developer.android.com/sdk/ndk/1.6_r1/index.html) for developing applications in Java and C/C++ as well as code samples.

WARNING*: *Please refer to the Known Issues section at the end of this document before booting your devkit. There are several items that can in specific cases cause the devkit to be unstable or not boot completely when running the current Android OS. Please consult the Known Issues section for fixes.

System Requirements

This pack requires additional hardware and software above and beyond the pack itself and the devkit as shipped. Additional accessories required beyond the ones listed in the next section are described in the NVIDIA Tegra 250 HW Setup Guide.

Throughout this document, it is assumed that the development host PC is running Microsoft Windows; Windows XP has been extensively tested, and newer versions such as Vista and Windows 7 should function as well. However, 64-bit variants of Microsoft Windows have not been tested. Please consult the NVIDIA Tegra Developers' website (http://developer.nvidia.com/tegra) , specifically the forums with any 64-bit development PC OS issues.

Android development on Mac OS X and Linux is supported; only minor deviations from these instructions should be required. However, only Windows XP has been tested with this release. Currently, while Windows, Linux and Mac OS X development PCs can be used to develop with a Tegra 250 devkit already flashed and booted to Android, *only Windows and Linux are supported operating systems for actually flashing the devkit initially*

Note: Currently, Mac OS X cannot be used to flash a Tegra 250 devkit. It can only be used to develop over ADB with a devkit already flashed using Linux or Windows.

Hardware

NVIDIA recommends

At least a ~2.0GHz CPU

1-2GB RAM

Software

Required

Tegra 250 Android Platform Support Pack

Download and install the latest installer for your development PC's operating system from http://developer.nvidia.com/tegra. For Windows development PCs, this is:

```
android_tegra_250_<version>.msi
```

For Linux development PCs, this is (the pack may be provided zipped or directly as an installable file)

```
android_tegra_250_<version>.run[.zip]
```

Android SDK

Download and install the Android SDK from
http://developer.android.com/sdk/index.html

Add the Android 2.0 SDK platform as described in
http://developer.android.com/sdk/adding-components.html

Add the Windows USB driver as described in
http://developer.android.com/sdk/win-usb.html
Additional changes as described later is in this document is required to the driver's inf file for it to install.

Required for C/C++ development

Cygwin

Download and install cygwin from http://www.cygwin.com.

Android NDK

Download and install the Android NDK from
http://developer.android.com/sdk/ndk/1.6_r1/index.html

Nice to have

Eclipse

Download and install the Eclipse IDE for Java Developers package from http://www.eclipse.org/downloads/

Android Development Tools (ADT) Plugin

This allows you to compile Android projects from within Eclipse.

Follow http://developer.android.com/sdk/eclipse-adt.html to install the ADT plugin.

C/C++ Development Tools (CDT) Plugin

This allows you to compile C/C++ code from within Eclipse.

Install CDT by following the same steps as when installing the ADT plugin but using the URL http://download.eclipse.org/tools/cdt/releases/galileo as the Available Software Site, and then selecting CDT Main Features -> Eclipse C/C++ Development Tools.

Environment variables (optional)

While not required to be set, these environment variables will be referred to throughout this document.

These environment variables should be set either by right clicking

My Computer->Properties->Advanced->Environment Variables

or directly in your Eclipse workspace via the menu item

Window->Preferences->C/C++->Environment.

If you set these via My Computer, you'll have to restart Eclipse for these settings to take effect.

NDKROOT

Set the environment variable NDKROOT to the installation path of the NDK. For example C:\android\android-ndk-1.6_r1

Windows specific

CYGWIN_HOME

Set the environment variable CYGWIN_HOME to the root of your Cygwin installation. For example c:\cygwin

Installing the Support Pack

On Windows

To install the Support Pack, simply double click the installation file

```
android_tegra_250_<version>.msi
```

End-User License

You will be prompted with End-User License Agreement. Read the agreement and to accept it check the box "I accept the terms in the License Agreement". Click next to proceed.

Installation Type

There are 3 ways to install the Support Pack. Read the instructions on the screen and choose one of the three options.

"Typical" and "Complete" setup options will install the Support Pack in

```
C:\Program Files\NVIDIA Corporation\android_tegra_250_<version>\
```

by default. If you want to install the Support Pack to a different location then choose the "Custom" setup option, change the installation location and click Next.

Install the new Platform Support Pack to a ***different*** directory tree than any other platform support packs. Overlapping platform packs in a single tree will lead to problems.

On Linux

To install the Support Pack, unzip it from a `.zip` to the contained `.run` (if the file was shipped zipped). Then, open up a terminal and run

```
sh android_tegra_250_<version>.run
```

End-User License

You will be prompted with End-User License Agreement. Read the agreement and to accept it type "yes" and press enter. The OS image and flashing utilities will be extracted to the directory

```
./android_tegra_250_<version>
```

Installing ("Flashing") Android to the Devkit

Once the Platform Support pack is installed to the host PC, it is possible to flash the included OS image to the devkit.

Prerequisites

Refer to the diagrams of the connections and the main board, as well as the instructions for putting the hardware into recovery mode in the Devkit HW Setup guide.

Selecting and Connecting the Desired Display

The Android OS image can support booting to VGA (15-pin D-Sub) or HDMI (HDMI also supports DVI-D via HDMI-to-DVI-D connectors). The selection of display device is currently an OS flash-time decision. Select your desired video-out option and connect the display to the corresponding jack.

For maximum compatibility, please ensure that your boot display is plugged in before powering on the devkit, so that the OS can detect the display properly during boot. Additionally, if you choose to use a Keyboard-Video-Mouse (KVM) switch to share display and input devices, you should have the devkit's input on the KVM switch selected and active prior to boot to ensure it can properly read the capabilities of the boot display.

Placing the devkit into Recovery ("Flashing") Mode

Refer to the NVIDIA Tegra 250 devkit HW Setup guide for details on how to put the devkit into recovery mode.

Flashing using Windows

The Android Platform Support pack includes recovery mode USB drivers in the directory

```
os/usbpcdriver
```

Once the display type is selected and the devkit is in recovery mode, the OS image can be flashed. The Platform Support pack includes two batch files in the "`os`" subdirectory that can be used to flash the OS:

Display Type	**Flash Batch Script**
CRT	**`nvflash_1gb_crt.bat`**
HDMI	**`nvflash_1gb_hdmi.bat`**

The flashing process will begin immediately. At the end of a successful flashing, the device will reboot to the desired video out mode with the Android desktop. If run from a command prompt (rather than double-clicking the batch file), the resulting output should be similar to the following:

```
Nvflash started
rcm version 0X20001
System Information:
   chip name: t20
   chip id: 0x20 major: 1 minor: 2
   chip sku: 0x8
   chip uid: 0x0808010541bfa0d7
   macrovision: disabled
   hdcp: enabled
   sbk burned: false
   dk burned: false
   boot device: nand
   operating mode: 3
   device config strap: 0
   device config fuse: 0
   sdram config strap: 0

sending file: tegra_250_333MHz_1GB.bct
- 4080/4080 bytes sent
tegra_250_333MHz_1GB.bct sent successfully
odm data: 0x300011
downloading bootloader -- load address: 0x108000 entry point: 0x108000
sending file: fastboot.bin
\ 884016/884016 bytes sent
fastboot.bin sent successfully
waiting for bootloader to initialize
bootloader downloaded successfully
setting device: 1 0
creating partition: BCT
creating partition: PT
creating partition: EBT
creating partition: UIP
creating partition: USP
creating partition: SOS
creating partition: LNX
creating partition: APP
creating partition: CAC
creating partition: UDA
Formatting partition 2 BCT please wait.. done!
Formatting partition 3 PT please wait.. done!
Formatting partition 4 EBT please wait.. done!
Formatting partition 5 UIP please wait.. done!
Formatting partition 6 USP please wait.. done!
Formatting partition 7 SOS please wait.. done!
Formatting partition 8 LNX please wait.. done!
Formatting partition 9 APP please wait.. done!
Formatting partition 10 CAC please wait.. done!
Formatting partition 11 UDA please wait.. done!
done!
sending file: fastboot.bin
\ 884016/884016 bytes sent
fastboot.bin sent successfully
sending file: flashboot.img
/ 2308096/2308096 bytes sent
flashboot.img sent successfully
sending file: system.img
/ 75989760/75989760 bytes sent
system.img sent successfully
```

Flashing using Linux

No special USB drivers are required to flash using Linux.

Once the display type is selected and the devkit is in recovery mode, the OS image can be flashed. The Platform Support pack includes two shell files in the directory of the OS image that can be used to flash the OS:

Display Type	Flash Batch Script
CRT	**nvflash_1gb_crt.sh**
HDMI	**nvflash_1gb_hdmi.sh**

To execute one of them, start a terminal and run (for example)

```
chmod +x nvflash_1gb_crt.sh
./nvflash_1gb_crt.sh
```

The flashing process will begin immediately. At the end of a successful flashing, the device will reboot to the desired video out mode with the Android desktop. The resulting output should be similar to the Windows version's output.

Making the Android Debug Bridge (ADB) work

Installing the Windows USB Driver

For the NVIDIA Tegra to be recognized by Google's Windows ADB drivers, the file

`android-sdk-windows\usb_driver\android_winusb.inf`

in the Android SDK must be edited to have the following added in the

`[Google.NTx86]`

section:

```
;NVIDIA Tegra
%SingleAdbInterface%        = USB_Install, USB\VID_0955&PID_7000
%CompositeAdbInterface%     = USB_Install, USB\VID_0955&PID_7100&MI_01
```

When the device is connected and the new hardware wizard pops up, pointing the wizard to the location of this modified android_winusb.inf will enable you to install the *Android Composite ADB Interface*.

Making ADB recognize the Tegra device

In addition to installing the USB driver, ADB must be configured to use our device. This is done by entering the following command in a Windows command prompt:

```
echo 0x955 >> "%USERPROFILE%\.android\adb_usb.ini"
```

Or on Linux and OS X with the following commands:

```
mkdir -p ~/.android
echo 0x955 >> "~/.android/adb_usb.ini"
```

Once this is done, restart adb with the "`adb kill-server`" command and makes sure "`adb devices`" lists the Tegra device.

> **Note:** If the adb connection hangs, try typing `adb kill-server` in a command prompt window to restart it.

Developing an application

Please refer to http://developer.android.com/guide/index.html for more information on developing on Android. A good place to start is the "Hello World" tutorial available at http://developer.android.com/resources/tutorials/hello-world.html.

Frequently asked questions

How do I compile Android C/C++ code inside Eclipse?

1) First create a new or load an existing Android project into Eclipse.
2) Select the menu item File->New->Other->C/C++->Convert to a C/C++ Project.
3) Select "Makefile project" and "— Other toolchain —", and click finish. This will allow you to build the C/C++ code using a makefile that makes use of the NDK compilers.
4) Right click your project and select properties

5) Under C/C++ Build, make sure the build directory is what you want. Typically for Android applications it'll be ${workspace_loc:/<myproject>/jni}

6) Under C/C++ Build->Environment, add a new variable with the name "PATH" and the value "${CYGWIN_HOME}/bin".

7) Under C/C++ General->Paths and symbols, add any directories you want to have scanned for code completion. For example *${NDKROOT}/build/platforms/android-4/arch-arm/usr/include* for the 1.6 NDK.

8) Now create a makefile in your project's jni directory. For inspiration, build an NDK app with the *V=1* parameter. For example:

```
make APP="hello-jni" V=1
```

This will show the command line used to create the .so. Make sure the dynamic library is output to the <myproject>/libs/armeabi directory.

9) Build the project by selecting the menu item Project->Build all. If all goes well the newly created makefile will have compiled your C/C++ code and the generated APK file will contain the .so.

My existing C/C++ based application won't compile and link!

The current Android NDK released by Google (1.6) only provides a limited set of native APIs and libraries for C/C++ development. For example, there is no C++ RTTI, exception or STL support. For more details on the limitations and what is supported, please read the following files provided with the NDK:

```
${NDKROOT}/docs/SYSTEM-ISSUES.TXT
${NDKROOT}/docs/STABLE-APIS.TXT
${NDKROOT}/docs/system/libc/OVERVIEW.TXT
```

While not tested by NVIDIA, if the STL functionality is required there are ports of STLPort and uSTL for Android available on the internet.

After exiting my application and launching it again, weird things happen…

As explained at http://developer.android.com/guide/topics/fundamentals.html#proclife, the process of an application can be kept alive as a cache to improve startup time of future launches. So even if your activity has exited, the process that it belongs to can still be alive. As static variables belong to the process and not the activity instance, as long as the process is alive these will not be de-allocated when the activity ends and will not be re-initialized when the

activity starts. To work around this, make sure to either not use static variables, or manually de-allocate and re-initialize them when appropriate.

Where can I find the USB drivers?

There are two different sets of USB drivers; the NVIDIA Recovery USB drivers required when flashing the device and the Windows ADB drivers used for the Android Debug Bridge. The recovery drivers are included with the Android OS flash pack. To install them, just point the installer to the directory of the recovery driver when the new hardware wizard shows up.

> We do not ship specific ADB drivers, but rather the official ones from Google are tweaked to recognize our device. This is described in the sections "System Requirements" and "Making the Android Debug Bridge (ADB)" work found earlier in this document.

Why is my app not full screen?

Check the AndroidManifest.xml file of your application. It should specify the minSdkVersion 4 or higher. If minSdkVersion is either not specified or if it is 3 or below then application will not be full screen. Such applications do not support higher resolutions. Use one of the following:

```
<uses-sdk android:minSdkVersion="5" />
<uses-sdk android:minSdkVersion="4" />
```

You can find related discussion in Android Developers Group:
http://groups.google.com/group/android-developers/browse_thread/thread/d5ac812e206a3cb7/6f6323b379411485?lnk=gst&q=Need+Help+%3A+getheight%28%29%2Fwidth%28%29+API+returning+&pli=1

Where can I get more information about Android Widgets?

In Android Widgets are used to display information about an application (like calendar events or song being played etc.) in views that can be embedded in other applications like the home screen. Please refer to the following for more information on how to write Widgets.

> http://developer.android.com/guide/topics/appwidgets/index.html
>
> http://developer.android.com/guide/practices/ui_guidelines/widget_design.html
>
> http://android-developers.blogspot.com/2009/04/introducing-home-screen-widgets-and.html

The Screen goes dark after 1 minute and comes up as the lock screen (normal Android behavior)

On a USB keyboard, use the Windows/Menu key or F1 to unlock the lock screen. Then on the device go to

Settings->Applications->Development

and toggle the "Stay awake" option. Alternatively, the screen timeout can be changed via

Settings->Sound & Display->Screen timeout.

Known Issues

This is an alpha release and is not representative of a final OS image.

ADB connection hangs

If the ADB connection hangs, start a command prompt and execute "adb kill-server" to make it restart.

Display is left shifted when the default boot display is CRT

Please try another monitor, use the auto-adjust menu item on the monitor (if available), manually adjust the monitor or use the HDMI output.

The device reboots constantly with some USB peripherals attached

Please try using another USB mouse. If the device is connected to the computer via a USB cable, please unplug and then reconnect when the device is fully booted.

If you connect *both* D-Sub (CRT) and HDMI to the devkit, HDMI will not work

Do not connect both HDMI display and CRT (VGA) displays to the devkit simultaneously; only connect the type selected as a part of the flashing process.

NVIDIA Corporation
2701 San Tomas Expressway
Santa Clara, CA 95050
www.nvidia.com