



# Android Oprofile profiling on the Tegra 200 series devkit

---

Version 0.1

---

# Contents

---

|              |   |
|--------------|---|
| INTRODUCTION | 3 |
| INSTALLATION | 3 |
| PROFILING    | 4 |

# Introduction

---

This document shows a quick overview of how to get oprofile profiling going on the Tegra development kit.

## Prerequisites

At the moment a Linux host computer set up for android development is required. On Ubuntu the following packages had to be installed to compile oprofile:

```
sudo apt-get install binutils-dev
sudo apt-get install libpopt-dev3
```

## Installation

---

1. Extract the oprofile helper scripts in your ~/android directory:

```
cd ~/android/
tar xvfz oprofile.tar.gz
cd oprofile
```

2. Download the latest oprofile-0.9.6:

```
wget http://prdownloads.sourceforge.net/oprofile/oprofile-0.9.6.tar.gz
```

3. Extract the package, run configure and make:

```
tar xvfz oprofile-0.9.6.tar.gz
cd oprofile-0.9.6
./configure
make
```

4. Prepare your oprofile directory:

```
export OPROFILE_EVENTS_DIR=~/.android/oprofile/oprofile-0.9.6/host
mkdir -p $OPROFILE_EVENTS_DIR/bin
mkdir -p $OPROFILE_EVENTS_DIR/abi
mkdir -p $OPROFILE_EVENTS_DIR/arm
cp ~/.android/oprofile/arm_abi $OPROFILE_EVENTS_DIR/abi
cp $OPROFILE_EVENTS_DIR/../../libabi/opimport $OPROFILE_EVENTS_DIR/bin
cp $OPROFILE_EVENTS_DIR/../../pp/opreport $OPROFILE_EVENTS_DIR/bin
cp -r $OPROFILE_EVENTS_DIR/../../events/arm/* $OPROFILE_EVENTS_DIR/arm
```

5. Flash the device with the oprofile enabled OS.

# Profiling

---

## Device setup

1. Setup your adb and get into adb shell.

```
adb remount
adb shell
```

2. Run oprofile setup on adb shell. Both opcontrol and oprofiled can be found under /system/xbin which should be in your \$PATH already. Using the CPU\_CYCLES event as an example:

```
opcontrol --setup
oprofiled --session-dir=/data/oprofile --no-vmlinux -e CPU_CYCLES
```

You should see the following message from oprofiled daemon:

```
Using 2.6+ OProfile kernel interface.
Reading module info.
Using log file /data/oprofile/samples/oprofiled.log
```

If, instead, "opcontrol --setup" says:

```
Cannot create directory /dev/oprofile: File exists
do_setup failed#
```

it's likely that your OS image does not have oprofile support built in. Please flash an OS image that does support it.

3. Next, control start and stop of samples collection while running tests or executing some applications.

```
opcontrol --start
...
Run your application
...
opcontrol --stop
```

4. The profiling data is now captured in /data/oprofile which needs to be pulled over to the host machine for post processing. You can always check the status and print the help information that list parameters:

```
opcontrol --status
opcontrol --help
oprofiled --help
```

5. OProfile will accumulate samples over several profiling runs. To clear earlier samples before starting a new run, you need to do

```
opcontrol --shutdown
opcontrol --reset
<rerun oprofiled>
```

## Host side post processing

Once you have run oprofile on the device, you are ready to run opimport\_pull script to generate a short summary of profiling results. The script requires a location where the sample data is to be located on the host machine, for example:

```
export OPROFILE_EVENTS_DIR=~/.android/oprofile/oprofile-0.9.6/host
cd ~/.android/oprofile
python -E opimport_pull -r /tmp/oprofile
```

## Advanced and experimental topics

### *Additional profiling info*

You can generate additional profiling result that will scan through the target modules and save into files. Add this line at the bottom of opimport\_pull:

```
os.system(oprofile_event_dir + "/bin/oproport --session-dir=. -p $OUT -l -t 0.1")
```

Set the OUT variable to the location of the built libraries on your local machine for symbol lookups.

This additional reporting will take extensive host cycles and you may see some warning messages regarding modules' timestamps.

### *Getting separate statistics for each CPU core*

Add --separate-cpu=1 to your oprofiled command line before running the profiling on the device and change line 63 of opimport\_pull to

```
stream = os.popen("find raw_samples -type f -name \*all.\*")
```

**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, EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE MATERIALS, AND EXPRESSLY DISCLAIMS ALL IMPLIED WARRANTIES OF NONINFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE.

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, GeForce, NVIDIA Quadro, and NVIDIA CUDA 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**

© 2008-2010 NVIDIA Corporation. All rights reserved.

**NVIDIA**

NVIDIA Corporation

2701 San Tomas Expressway

Santa Clara, CA 95050

[www.nvidia.com](http://www.nvidia.com)