

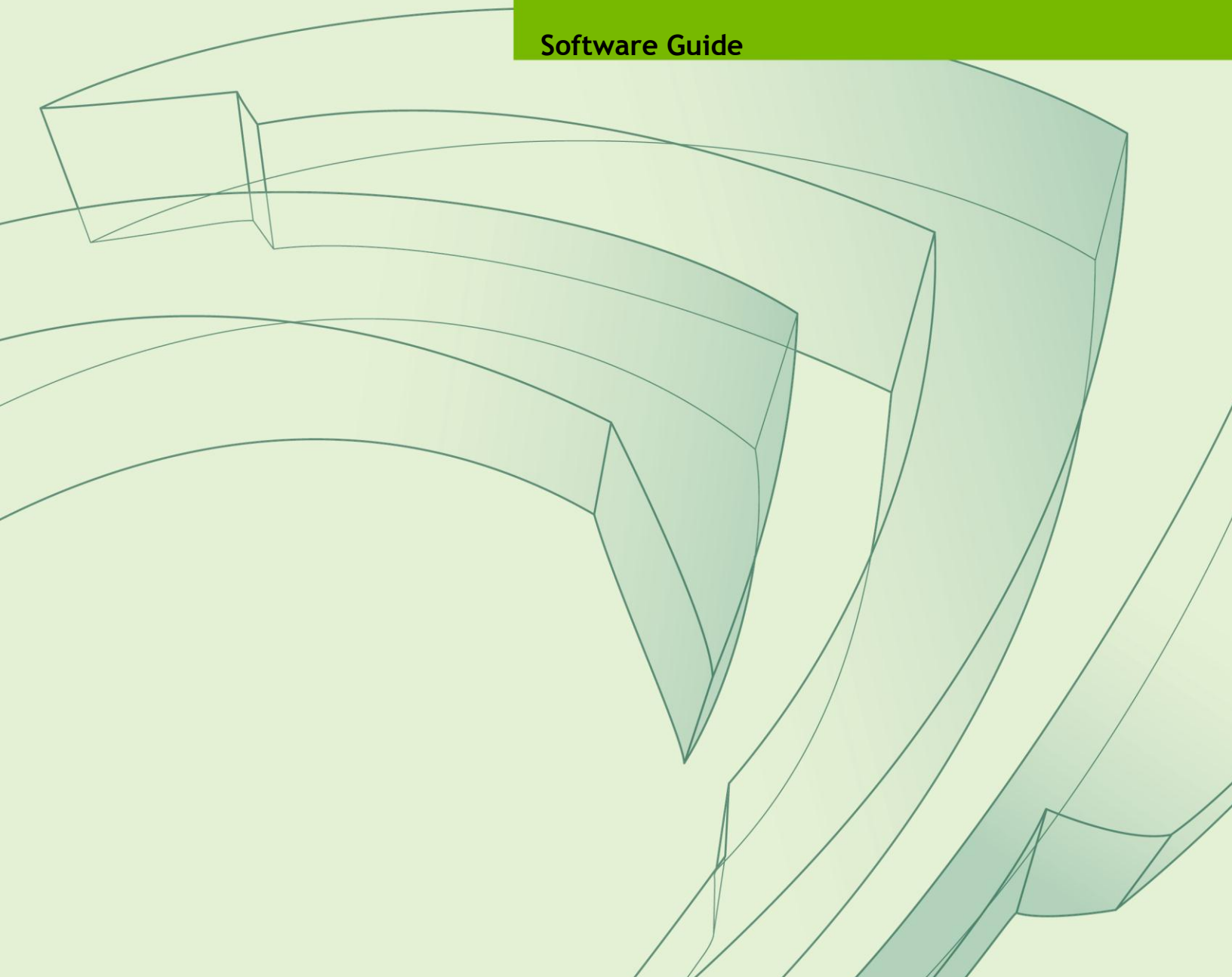


# NVIDIA

## FIELD DIAGNOSTIC SOFTWARE

DU-05363-001\_v20 | January 2015  
NVIDIA CONFIDENTIAL | Prepared and Provided Under NDA

### Software Guide



## DOCUMENT CHANGE HISTORY

DU-05363-001\_v20

Version	Date	Authors	Description of Change
01	June 19, 2010	CG	Initial Release
02	August 16, 2010	CC	Added gen1, x8, and gpu_temp arguments.
03	August 23, 2010	CC	Added Linux section. Clarified DOS and Linux system requirements
04	September 10, 2010	CC	Added new supported Tesla products Updated the maximum tested Linux kernel version
05	October 11, 2010	CC	Added Tesla X2070 Added optional command line arguments
06	October 13, 2010	CC	Corrected “device” argument callout. Removed “-v” argument.
08	November 2, 2010	CC	Added gpu_temp=ipmi: option.
09	November 22, 2010	CC	Added “p0only” argument.
10	March 23, 2011	CC	Added Tesla M2090/X2090 products and revised command line argument table.
11	June 15, 2011	CC	Added Tesla M2075/C2075 products and updated command line argument table.
12	July 18, 2012	CC	Added Tesla K10 products and updated command line argument table.
13	August 30, 2012	CC	Added Tesla K20 products.
14	October 5, 2012	CC	Added log file option.
15	December 12, 2012	CC	Added enable_graphics and poll_interrupts options.
16	February 11, 2013	CC	Added supported TinyLinux version, and <a href="#">Section 3.3 Running an Upgraded Field Diag Version</a> .
17	December 9, 2013	CC	Added support for Tesla K40. Updated <a href="#">Section 2.0 USAGE</a> . Added power_cap_limit and disable_progress_bar arguments.

Version	Date	Authors	Description of Change
18	May 1, 2014	CC	Added support for Tesla K8.
19	November 4, 2014	CC	Added support for Tesla K40d and Tesla K80
20	January 22, 2015	CC	Added pciid argument.

## TABLE OF CONTENTS

<b>The NVIDIA Field Diagnostic Software .....</b>	<b>1</b>
1.0 Introduction .....	1
2.0 Usage .....	2
2.1 System Requirements .....	2
Linux .....	2
2.2 Test Progress .....	3
2.3 Return Codes .....	3
2.4 Error Logs .....	4
2.4 Field Diagnostics Command-Line Arguments .....	5
2.4.1 Argument List .....	5
2.4.2 Additional Instructions .....	8
3.0 Running Under Linux .....	9
3.1 Prerequisites for Running Under Linux .....	9
Kernel Version .....	9
Kernel Architecture .....	9
glibc Version .....	9
Installing the Field Diagnostic Kernel Module .....	10
Do Not Run the NVIDIA Kernel Module .....	10
Unload the Nouveau Driver .....	10
Disable any Framebuffer Consoles .....	10
3.2 Installing the Kernel Module .....	11
3.3 Running an Upgraded Field Diag Version .....	12
4.0 GPU Tests .....	12

# THE NVIDIA FIELD DIAGNOSTIC SOFTWARE

The information in this document is confidential and is the property of NVIDIA Corporation. This document may not be distributed without prior NVIDIA authorization.

## 1.0 INTRODUCTION

This document describes the NVIDIA Field Diagnostic. This is a powerful software program that allows users to test the following NVIDIA Tesla hardware:

- ▶ Tesla C2050, C2070, C2075
- ▶ Tesla M2050, M2070, M2070Q, M2075
- ▶ Tesla S2050
- ▶ Tesla X2070, X2090
- ▶ Tesla M2090
- ▶ Tesla K8, K10, K20, K20X, K40, K40d, K80

The NVIDIA Field Diagnostic runs on most Linux variants based on a 2.6 kernel.

## 2.0 USAGE

### 2.1 System Requirements

#### Linux

- ▶ Intel Pentium III or later CPU
- ▶ 2GB or more of system memory.
- ▶ Linux kernel 2.6.16 or later

Kernel 2.6.29 or later is recommended for performance reasons. The tool has been tested with kernels 2.6.16 through 2.6.35.

Linux 2.4 is not supported.

- ▶ glibc 2.3.2 or later
- ▶ 64-bit x86\_64 kernels are supported. 32-bit kernels are not supported.
- ▶ TinyLinux version 12.08 or later is supported.

## 2.2 Test Progress

When the diagnostic is running, it displays the test progress on the screen in the following format:

```
"Running test X on GPU n - Y tests remaining  |=====  | Z %"
```

Where:

**X** is the current test number in the sequence

**n** is the GPU number

**Y** is the number of remaining tests

**Z** is the overall percent completion

Example:

```
Running test 208 on GPU 0 - 118 tests remaining  |=====  | 64 %
```



**Note:** You can disable the test progress display by using the command line option `disable_progress_bar`.

## 2.3 Return Codes

When completed, the diagnostic will return 0 to the shell if completed normally. If an error occurs, it will return 1 to the shell, and if a retest is required it will return 2. It will also print "PASS", "FAIL", or "RETEST" to the screen.

- ▶ **PASS** – The hardware passes the diagnostics
- ▶ **FAIL** – The hardware has failed the diagnostics
- ▶ **RETEST** – The hardware setup has failed the pre-check portion of the diagnostics and a warning message appears describing the problem. Correct the problem per the pre-check message and then test again.

## 2.4 Error Logs

By default, Field Diagnostics produces a binary log file which is used by NVIDIA engineers to diagnose a failing card. The name of the log file incorporates basic test result information as well as the serial number of the board under test:

`fielddiag_<PASS/FAIL/CONFIG>_<serial number>.log`

- Example log file name for a passing board: **“fielddiag\_PASS\_1234567.log”**.
  - Example log file name for a failing board: **“fielddiag\_FAIL\_1234567.log”**.
  - Example log file name for a board that failed the initial SKU configuration: **“fielddiag\_CONFIG\_1234567.log”**.
- If multiple cards in the system are tested, the serial number of the first GPU is incorporated into the name of the log file.
- If a problem prevents reading the serial number from the card, the returned file name will not contain the serial number.



**Note:** You can override the default log file name by using the command line option **logfile**.

## 2.4 Field Diagnostics Command-Line Arguments

Normally, the field diagnostic is invoked by using the command-line:

- `fieldiag` (to test all GPUs found in the system)

### 2.4.1 Argument List

Table 1 lists the optional command line arguments and provides a brief description of each. Further explanation is given in the next section.

Table 1. Optional Command Line Arguments

Option	Description	Application
<code>device=&lt;n&gt;</code>	Test the <i>n</i> th device.	All
<code>gen1</code>	Run <code>fieldiag</code> compliant with Gen1 PCIe.	All
<code>gen2</code>	Run <code>fieldiag</code> compliant with Gen2 PCIe.	All
<code>x8</code>	Run <code>fieldiag</code> compliant with x8 PCIe.	All
<code>volterra_disable</code>	Disable thermal monitoring of regulator slave temperatures.	Tesla X2070 Tesla X2090
<code>gpu_temp=[ext,ADT7473,ADT7461,canoas,disabled,ipmi,int]</code>	Use the specified sensor for acquiring the GPU temperature:	
	<b>gpu_temp=ext:</b> Read the GPU temperature using an external temperature chip that communicates back to the GPU.	Tesla C2050 Tesla C2070 Tesla C2075
	<b>gpu_temp=ADT7473:</b> Read the GPU temperature from the ADT7473 over a supported motherboard SMBus.	Tesla M2050 Tesla M2070 Tesla M2075 Tesla M2090
	<b>gpu_temp=ADT7461:</b> Read the GPU temperature from the ADT7461 over a supported motherboard SMBus.	Tesla X2070 Tesla X2090
	<b>gpu_temp=canoas:</b> Read the GPU temperature from the system microcontroller.	Tesla S2050

	<b>gpu_temp=disabled:</b> Disable temperature logging if motherboard SMBus support is not available and results in fielddiag test failures.	Tesla M2050 Tesla M2070 Tesla M2075 Tesla X2070 Tesla M2090 Tesla X2090
	<b>gpu_temp=ipmi:&lt;id0&gt;:...:&lt;idn&gt;</b> Read the GPU temperature over the IPMI bus for each GPU in the system, where <idx> refers to each GPU's sensor ID (colon separated). Ex: <id0> is the sensor ID for GPU0, etc. To specify one GPU in a multi-GPU system when testing with device=<n>, populate non-tested ids with "0". Ex: gpu_temp=ipmi:0:0:0<id3> for GPU3. When not testing with device=<n>, a valid GPU sensor ID must be provided for all connected GPUs.	Tesla M2070 Tesla M2075 Tesla M2090 Linux
	<b>gpu_temp=int:</b> Read the temperature using the internal GPU sensor circuit for boards that support internal temperature reading.	Tesla K8 Tesla K10 Tesla K20(X) Tesla K40 Tesla K40d Tesla K80
p0only	Run the field diagnostic only in PState 0.	Tesla C2050 Tesla C2070 Tesla C2075 Tesla M2070 Tesla M2075 Tesla X2070 Tesla M2090 Tesla X2090 Tesla K8 Tesla K10 Tesla K20(X) Tesla K40 Tesla K40d Tesla K80
logfile=<filename path>	Specify a unique log file name other than the default. Ex. logfile=/var/log/mylogfile.log	All

enable_graphics	Enable graphics capability (useful for additional test coverage). See requirements for using this option in section <a href="#">2.4.2 Additional Instructions</a> .	Tesla K20(X)
poll_interrupts	Use CPU polling for handling Tesla card interrupts.	All
power_cap_limit=<X>	Specify the limit, in watts, at which the board power consumption will be capped. Ex. <b>power_cap_limit=200</b> (Limit power to 200 watts)	Tesla K8 Tesla K10 Tesla K20(X) Tesla K40 Tesla K40d Tesla K80 Linux
disable_progress_bar	Prevents the test progress from being displayed on the screen (see <a href="#">2.2 Test Progress</a> ).	All
pciid=<w:x:y.z>	Specify the board to be tested, where w, x, y, z are hexadecimal numbers: <b>w</b> is the PCI domain (must be present, but only zero is supported; any other value causes fielddiag to fail) <b>x</b> is the PCI bus (basically, which PCIe slot the board is installed in) <b>y</b> is the device <b>z</b> is the function  Ex. <b>pciid=0:2:0.0</b> See <a href="#">2.4.2 Additional Instructions</a> for more information.	All

## 2.4.2 Additional Instructions

This section provides addition instructions for specific command line arguments.

### enable\_graphics

To run the Field Diagnostic with graphics mode enabled, the field diagnostic must be run directly after a reboot cycle without the NVIDIA driver loaded. Additionally, after the Field Diagnostic is completed, a reboot is required to restore the card to normal operation.

### pciid

- This command line argument can appear only once on the command line.
- The device command line argument cannot appear in the same command line.

Use nvidia-smi to obtain the values to enter. For example:

```
# nvidia-smi -q

=====NVSMI LOG=====

Timestamp                  : Tue Jan 20 18:23:18 2015
Driver Version             : 340.65

Attached GPUs              : 3
...
GPU 0000:01:00.0
  Product Name              : Tesla K40c
...
  PCI
    Bus                     : 0x03
    Device                  : 0x00
    Domain                  : 0x0000
    Device Id               : 0x102410DE
    Bus Id                  : 0000:01:00.0
```

In this example, obtain the values from either “Bus Id” or “GPU” => 0000:01:00.0.

The command line argument would then be **pciid=0:1:0.0** .

## 3.0 RUNNING UNDER LINUX

This section applies to users who wish to run the Field Diagnostic on a Linux distribution other than the one provided by NVIDIA. If you are using the NVIDIA-supplied distribution you can skip this section.

### 3.1 Prerequisites for Running Under Linux

#### Kernel Version

Linux manufacturing Field Diagnostics requires a minimum kernel version of 2.6.16. Version 2.6.29 or later is recommended for performance reasons. Older versions have not been tested and may not work. Kernel 2.4 is not supported. The version of the running kernel can be established by running:

```
$ uname -r
```

#### Kernel Architecture

Linux manufacturing Field Diagnostics is a 64-bit application and requires kernel compiled for x86\_64 architecture. To determine kernel architecture, type:

```
$ uname -m
```

#### glibc Version

The system on which Field Diagnostics is run must be built on glibc-2.3.2 or newer. To determine glibc version, type:

```
$ /lib/libc.so.6
```

## Installing the Field Diagnostic Kernel Module

Linux manufacturing Field Diagnostics includes a kernel module. The purpose of this module is to expose certain kernel-mode APIs to Field Diagnostics, which runs as a user-mode application. In order to be able to install the kernel module, the system must contain configured kernel sources and development tools, including make and gcc. Without them it is not possible to compile the kernel module. Use package manager provided by your distribution to install kernel sources. Typically the package name is kernel-sources or linux-sources. For example, on Debian type:

```
$ sudo apt-get install linux-source-`uname -r`
```

If you run Field Diagnostics as root, it will automatically run the included install\_module.sh script to compile and insert the Field Diagnostics kernel module. However if Field Diagnostics is not run by the root user, it is necessary to install the kernel module, which is recommended.

## Do Not Run the NVIDIA Kernel Module

For successful Field Diagnostics runs the NVIDIA GPUs in the system must be in their original, unaltered state, as initialized by the VBIOS. This means X must not have been run on the NVIDIA GPUs prior to running Field Diagnostics. Make absolutely sure that the NVIDIA kernel module is not loaded, otherwise the system may become unstable. In order to unload the NVIDIA kernel module it is necessary to first kill X. Killing X is also recommended even if it is using vesa or fb driver.

To disable X in SuSE, disable the xdm service in YaST.

On Debian-based systems (including Ubuntu), type:

```
$ sudo update-rc.d -f gdm remove
```

If not using gnome, type kde or xdm instead of gdm (as applicable).

## Unload the Nouveau Driver

Some newer Linux distributions include the nouveau driver in the kernel. This driver performs a kernel mode set and it also supports a framebuffer console. For Field Diagnostics to function correctly, this driver has to be unloaded (preferably blacklisted) so that it is not automatically loaded at boot.

## Disable any Framebuffer Consoles

Framebuffer consoles are also not recommended, because they may modify memory of the tested device during the tests. To disable the framebuffer console, edit /boot/grub/menu.lst and make sure the kernel arguments contain vga=normal instead of any other value. Make sure they do not contain anything like video=.

## 3.2 Installing the Kernel Module

Linux Field Diagnostics relies on a kernel driver to handle cases where it is necessary to use kernel-mode APIs. The easiest way to install the Field Diagnostics kernel module is to use the provided installation script, which you will find in Field Diagnostics runspace:

```
$ ./install_module.sh --install
```

[Note: You can't install the kernel module from a network directory where the root user does not have write access. In this case copy `install_module.sh` and `driver.tgz` to `/tmp` and run it there.]

This script also creates a udev configuration file in `/etc/udev/rules.d/99-mods.rules`. This file specifies group which will be able to access the kernel module. By default this is the video group, such as for the NVIDIA driver. Make sure your user is in this group or change the group in the `99-mods.rules` file to match one of yours. On some systems, the install script created file `/etc/udev/permissions.d/99-mods.permissions` instead, which simply lists user, group, and mode for the driver file.

To find out which group the driver has been assigned to, type:

```
$ ls -l /dev/mods
```

```
crw-rw---- 1 root video 10, 60 Jan 7 08:21 /dev/mods
```

To find out which groups you are in, type:

```
$ id
```

```
uid=4384(modsuser) gid=30(hardware) groups=30(hardware),1606(gpu-tesla)
```

If you decide to modify the group in `99-mods.rules`, you have to reload the kernel module:

```
$ modprobe -r mods
```

```
$ modprobe mods
```

To make sure the kernel module is always loaded when the system starts up follow your distribution-specific guidelines.

On Debian-based distros (such as Ubuntu) add the mods module name to `/etc/modules` if the installation script didn't add it.

On SuSE-based distros add the mods module name to `/etc/sysconfig/kernel` file in the `MODULES_LOADED_ON_BOOT` variable.

On RedHat-based distros (such as CentOS) add line `modprobe mods` to `/etc/rc.d/rc.local`.

## 3.3 Running an Upgraded Field Diag Version

If you have run an older version of the Field Diagnostic on your Linux distribution, the older MODS kernel module will conflict with the newer Field Diag and cause failures.

Before running a newer Field Diag, remove the older MODS kernel module as follows:

1. Navigate to the Field Diag subdirectory.
2. From the command line, run

```
./install_module.sh -u
```

When you run the later Field Diag, the updated MODS kernel module will be installed automatically.

## 4.0 GPU TESTS

The various tests in the package provide confirmation of the numerical processing engines in the GPU, integrity of data transfers to and from the GPU, and test coverage of the full onboard memory address space that is available to CUDA programs.

A typical GPU test performs the following operations:

1. Disable the windowing system to take over the entire screen.
2. Set the display mode and refresh rate.
3. Loop N times:
  - a. Exercise some aspect of the graphics hardware.
  - b. Read back the resulting image.
  - c. Calculate a 32-bit CRC, or possibly a checksum, to compare against the known correct value (golden value) for this GPU version and platform. For video and cursor tests use the hardware DAC CRC.
  - d. If the golden values do not match, report an error and abort the loop.
4. Restore previous display mode and refresh rate.
5. Release screen to the operating system.
6. Report test status.

Each test carefully chooses the random test parameters, i.e. invalid values are avoided, edge cases are properly covered, and proper weighting is given to more common cases.

## **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 of otherwise under any patent rights of NVIDIA Corporation. Specifications mentioned in this publication are subject to change without notice. This publication supersedes and replaces all other information previously supplied. NVIDIA Corporation products are not authorized as critical components in life support devices or systems without express written approval of NVIDIA Corporation.

## **Trademarks**

NVIDIA and the NVIDIA logo are trademarks or registered trademarks of NVIDIA Corporation in the U.S. and other countries. Other company and product names may be trademarks of the respective companies with which they are associated.

## **Copyright**

© 2010, 2015 NVIDIA Corporation. All rights reserved.