HW FIELD DIAG

vR331 | March 2014

HW Field Diag

THE NVIDIA FIELD DIAGNOSTIC SOFTWARE

1.1. 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 K10, K20, K20X, K40

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

1.2. 2.0 USAGE

1.2.1. 2.1 System Requirements

1.2.1.1. 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.

1.2.2. 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.

1.2.3. 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.

1.2.4. 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:

fieldiag_<PASS/FAIL/CONFIG>_<serial number>.log

Example log file name for a passing board: "fieldiag_PASS_1234567.log".
Example log file name for a failing board: "fieldiag_FAIL_1234567.log".
Example log file name for a board that failed the initial SKU configuration: "fieldiag_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 logfilename.

1.2.5. 2.4 Command-Line Arguments to Field Diagnostics

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

fieldiag (to test all GPUs found in the system)

1.2.6. 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.

Option	Description	Application
device=<n>	Test the nth device.	All
gen1	Run fieldiag compliant with Gen1 PCIE.	All
gen2	Run fieldiag compliant with Gen2 PCIE	All
x8	Run fieldiag compliant with x8 PCIE.	All
volterra_disable	Disable thermal monitoring of regulator slave temperatures.	Tesla X2070 Tesla X2090
gpu_temp=[ext,ADT7473,ADT7461, canoas,disabled,ipmi,int]	Use the specified sensor for acquiring the GPU temperature:	
	gpu_temp=ext: Read the GPU temperature using an external temperature chip that communicates back to the GPU.	Tesla C2050 Tesla C2070 Tesla C2075
	gpu_temp=ADT7473: Read the GPU temperature from the ADT7473 over a supported motherboard SMBus.	Tesla M2050 Tesla M2070 Tesla M2075 Tesla M2090
	gpu_temp=ADT7461: Read the GPU temperature from the ADT7461 over a supported motherboard SMBus.	Tesla X2070 Tesla X2090
	gpu_temp=canoas: Read the GPU temperature from the system microcontroller.	Tesla S2050
	gpu_temp=disabled: Disable temperature logging if motherboard SMBus support is not available and results in fieldiag test failures.	Tesla M2050 Tesla M2070 Tesla M2075 Tesla X2070 Tesla M2090 Tesla X2090

	gpu_temp=ipmi:<id0>:…:<idn> 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
	gpu_temp=int: Read the temperature using the internal GPU sensor circuit for boards that support internal temperature reading.	Tesla K10Tesla K20(X)
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 K10Tesla K20(X)
logfilename=<filename path>	Specify a unique log file name other than the default. Ex. logfilename=/var/log/mylogfile.log	All
enable_graphics	Enable graphics capability (useful for additional test coverage). See requirements for using this option in section 2.4.2 Additional Instructions.	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. power_cap_limit=200(Limit power to 200 watts)	Tesla K10Tesla K20(X)Tesla K40Linux
disable_progress_bar	Prevents the test progress from being displayed on the screen (see 2.2 Test Progress).	All

1.2.6.1. 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.

1.3. 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.

1.3.1. 3.1 Prerequisites for Running Under Linux

1.3.1.1. 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
```

1.3.1.2. 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
```

1.3.1.3. 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
```

1.3.1.4. 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.

1.3.1.5. 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 gdb (as applicable).

1.3.1.6. 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.

1.3.1.7. 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=.

1.3.2. 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.

1.3.3. 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:

Navigate to the Field Diag subdirectory.
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.

1.4. 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:

Disable the windowing system to take over the entire screen.
Set the display mode and refresh rate.
Loop N times:
Exercise some aspect of the graphics hardware.
Read back the resulting image.

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.

If the golden values do not match, report an error and abort the loop.

Restore previous display mode and refresh rate.

Release screen to the operating system.

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.