MODS
MODULAR DIAGNOSTIC SOFTWARE FOR 367.X DIAGNOSTICS

MODS.DOCX_R367_v02 | July 2015
NVIDIA CONFIDENTIAL | Prepared and Provided Under NDA

Software Documentation

DOCUMENT CHANGE HISTORY

MODS.DOCX_R367_v02

Version	Date	Authors	Description of Change
01	7/9/2015	Henry Wu	Initial Release of R361 mods.docx
02	3/14/2016	Stewart Reive	Updated Section 5.1 Error Codes

TABLE OF CONTENTS

LIST OF TABLES

MODULAR DIAGNOSTIC SOFTWARE (MODS)

1.0 INTRODUCTION

This document describes the NVIDIA Modular Diagnostic Software (MODS). MODS is a powerful software program that allows users to test NVIDIA hardware. MODS is used for three primary purposes:

- Chip and board functional validation
- Chip and board failure analysis and debug.
- Architectural verification

This document covers the usage of MODS for graphics and compute products.

GPU MODS is currently supported under the following operating systems

- Linux (2.6.18 kernel)
- Microsoft Windows 7
- MacOSX

MODS has the following features

- Embedded JavaScript (version 1.7)
- All of the low-level functionality exposed to the scripting language.

- Failure analysis and debug functionality is included — reading and writing of registers, memory, PIO, and PCI address spaces, clock programming, etc.
- One script will run on all supported operating systems without modification.
- Complete embedded OpenGL and CUDA drivers, and resource manager—this is the same code base that is used in the Linux and Windows drivers.

The MODS GPU manufacturing test suite exercises most but not all of the capabilities of the NVIDIA hardware. It is assumed that the silicon has undergone a normal screening process prior to shipping to the customer and that the primary purpose of the test is to determine if the board manufacturing process has completed successfully and all solder connections and components are working properly.

1.1 Notice to Users

NVIDIA has discontinued the support of DOS since R290. Please contact your NVIDIA representative about moving to Linux MODS.

2.0 USAGE

Normally, MODS is invoked by using the command-line:

- mods gputest.js –mfg (for CEM testing)
- mods gputest.js –oqa (for OEM outgoing QA testing)

The difference between these two test options is that the –mfg option runs the full suite of tests. The –oqa test is a slightly less stressful and quicker suite of the tests optimized for speed and coverage.

MODS test suite is usually distributed to customers in a package with a part number like "618-60506-3501-CX0." These packages have been qualified to test a particular product and contain release notes and batch files tailored to that card. The directions in those release notes should be followed instead of running the command-lines above.

2.1 Normal Usage

Usage:

- mods [options] [file] [JavaScript arguments]

Note that there are two types of options in MODS: those that are arguments to MODS itself, and those that are arguments to the script. By convention, MODS' arguments are usually a single character, but the script arguments are usually many characters.

Example:

- mods –d –C gputest.js –mfg –run_on_error

In the above example, -d and –C are optional arguments to MODS, and –mfg and –run_on_error are arguments to the script. For more optional arguments to MODS, please see section 3.3. For JavaScript based arguments (script dependent), please see section 3.4.

2.2 Interactive Mode

MODS has an interactive mode, which can be invoked with mods –s. This is a useful tool for debugging problems, but its use is beyond the scope of this document. To exit interactive mode, type "Exit()" or "q()".

2.3 Return Codes

MODS will return 0 to the shell under normal operation. If an error occurs, MODS will return non-zero error code to the shell. On Windows 7, MODS will also set the "MODS_EC" environment variable to the error code.

2.4 Error Logs

By default, MODS produces a human-readable log file named "mods.log." In presence of a "-json" script argument, mods also produces a "mods.jso" log file. The mods.jso is another version of the same data expressed in a markup language called JSON. (JSON is short for JavaScript Object Notation.) This second file is written to make it easier for automated upstream tools to analyze the result of the run.

3.0 DISTRIBUTION PACKAGE

Normally, MODS is invoked by using the command-line:

mods gputest.js –mfg

MODS is distributed with the following files:

Table 1. Files distributed with MODS

File	Description
cuda.bin default.bin msdec2_e.bin msdec2_m.bin msdec4_l.bin msdec4_s.bin msdec_pi.bin msdec_v.bin vic_data.bin vp2_stre.bin	Binary files used by various tests.
mats	Stand-alone memory test on Linux only. See section 7.0 of this document for more information.
mods	The main MODS binary.
cur_comm.he dev_p358.he drf.he fpk_comm.he glr_comm.he mods.he	Precompiled JavaScript header files.
arghndlr.jse boards.jse boostbase.jse comnargs.jse comngpu.jse comnmcp.jse comnmods.jse comnprnt.jse comntest.jse cudatest.jse dprun.jse edid.jse fileid.jse glrandom.jse gpuargs.jse gpudma.jse gpulist.jse gputest.jse gshmoo.jse intrutil.jse jsthread.jse mods.jse mods_eng.jse p358.jse prntutil.jse pstate.jse pvstest.jse random2d.jse shmclass.jse shmoo.jse	Precompiled JavaScript files.

testlist.jse thermcal.jse tofile.jse tunetrim.jse tunevolt.jse boards.dbe	
mods.pdf	This document
relnotes.txt	Release notes. These are updated with every MODS release.
gp100_f.jsone	Precompiled JSON files containing GPU-specific information.

3.1 Version Information

The MODS version may be obtained by running the following command.

- mods –v

The version is in the following format XX.YY where XX is the major version number, and YY is the minor version number. MODS uses NVIDIA's "unified software architecture" and much of the code base is shared with the drivers. A version of MODS with the version XX.YY (e.g., 343.5) has a lot of shared code with a driver that also starts with XX (e.g., 343.10).

3.2 System Requirements

Linux

- Intel or AMD CPU with AMD64 support
- 4GB or more of system memory
- Linux kernel 2.6.18 or newer
- GNU C library version 2.5 or newer

Apple Macintosh

- x86-based Macintosh. PowerPC-based systems are no longer supported.
- 4GB or more of system memory.

3.3 Command-line arguments to MODS

Table 2. MODS command line arguments

Option	Description
-a	append to log file
-c reference	display reference
-D	writes ‘debug’ level output to debug.log
-d	set 'debug' level output
-e script	execute script
-F string	set a filter for serial and circular sinks
-g	do not log return codes
-h or -?	print help
-i file	import JavaScript file
-l file	log file name; do not log if file is 'null'
-L	only write the log file if there is an error
-m script	execute script before main()
-n script	execute script after main()
-o	do not run main()
-P	enable circular buffer, set to 'debug' level & dump on exit
-r	record user input

-redir file	redirect standard output to file
-R	remote user interface (run over network)
-s	script user interface
-S level	enable serial sink and set it its level (from 1 to 4)
-t	macro user interface
-T	remote terminal user interface (telnet)
-U ip port	remote terminal user interface (client mode)
-w	raw user interface
-v	print MODS version
@<filename>	fetch command line arguments from <filename>

3.4 Command-line arguments for gputest.js

If no script file is specified, mods.js is used. If no log file is specified, mods.log is used. MODS parses the specified script file and any imported script files, and then executes the script method main(). You may optionally specify begin() and end() methods that are guaranteed to be called before and after main(), respectively.

Table 3. Options to gputest.js

Option	Description
--args <string>	Display Help for a particular subsection
-?	Display Help
-add <string>	Add the specified test(s).
-allow_ot_events <string>	Ignore this many overtemp events.
-arg <string>	Pass to wrapper scripts
-asr_enable <string>	Enable or disable ASR. 0=disable, 1=enable
-assume_battery_power	tell RM that we are on battery power.
-attrcb_timeslice_flag <string>	enable/disable attribute CB timeslice mode
-aza_maxsinglewaittime <string>	Set the Azalia maximum time to wait at a single time for simulation
-aza_timescaler <string>	Set the Azalia Timescaler for Simulation
-begin_dump_addr <string>	Start BAR0 address to log.
-bg_ext_temp <string> <string>	Start bg external therm sensor monitor and set print and read intervals
-bg_ext_temp_flush <string> <string>	Start bg external therm sensor monitor and set print and read intervals, flushing to disk after every print

-bg_fan <string> <string>	Start bg fan RPM monitor and sets print and read intervals
-bg_int_temp <string> <string>	Start bg internal therm sensor monitor and sets print and read intervals
-bg_int_temp_flush <string> <string>	Start bg internal therm sensor and sets print and read intervals, flushing to disk after every print
-bg_ipmi_temp <string> <string>	Start bg ipmi therm sensor monitor and sets print and read intervals
-bg_ipmi_temp_flush <string> <string>	Start bg ipmi therm sensor monitor and sets print and read intervals, flushing to disk after every print
-bg_power <string> <string>	Start bg power monitor and sets print and read intervals
-bg_power_flush <string> <string>	Start bg power monitor and sets print and read intervals, flushing to disk after every print
-bg_smbus_temp <string> <string>	Start bg smbus therm sensor monitor and sets print and read intervals
-bg_smbus_temp_flush <string> <string>	Start bg smbus therm sensor monitor and sets print and read intervals, flushing to disk after every print
-bg_tsosc <string> <string> <string> <string> <string> <string>	Poll tsosc speedo [countSel] [clks per meas] [outDiv] [adj] [print interval] [read interval].
-bg_volterra <string> <string>	Start Background Thermal monitor (including Volterra slave devices) and sets print and read intervals
-bg_volterra_flush <string> <string>	Start Background Thermal monitor (including Volterra slave devices) and sets print and read intervals, flushing to disk after every print
-bgdev	Start background GLStress test on all other devices.
-bgfunc <string>	Start the given function as a background there.
-bgstress	Start the background 3d stress task.
-bgtemp <string> <string>	Start Background Thermal monitor and sets print and read intervals
-bgtemp_flush <string> <string>	Start Background Thermal monitor and sets print and read intervals, flushing to disk after every print
-bgtest <string>	Loop this test number in a background thread.

-bgtest_flags <string> <string>	Run a bgtest with various flags, separated by a ','
-bgvolt <string> <string> <string> <string>	Log voltage droop [fbp/gpc/sys] [clks per meas] [print interval] [read interva].
-blacklist_pages_on_error	Blacklists bad physical address pages if memory tests hit an error.
-blcg	Block Level Clock Gating
-blcg2 <string>	Block Level Clock Gating.
-blcgIdleCGEnable	Block Level Clock Gating to enable IdleCG
-blcg_idle	Force BLCG Idle settings
-blcg_off	Block Level Clock Gating Off.
-blcg_quiescent	Force BLCG Quiescent settings
-blcg_stall	Force BLCG Stall settings
-blink_lights	Blink lights (keyboard LEDs for example) to show mods is not hung.
-boot_0_strap	Set Boot 0 strap
-boot_3_strap	Set Boot 3 strap
-bus_width <string>	Test for specific framebuffer bus width
-check_display	Run the CheckDisplay test.
-check_display_bar	Run the CheckDisplayBar test.
-check_display_bars	Run the CheckDisplayBars test.
-check_displays	Run the CheckDisplay test on all displays.
-check_driver_ver <string>	Check whether particular version of MODS kernel driver is installed
-check_ecc_on_init	Check ECC errors on init and fail if any exist
-check_features <string>	Test for a specific set of feature bits (3 args for 96 bits).

-check_features3 <string> <string> <string>	Test for a specific set of feature bits (3 args for 96 bits).
-check_fp_gray	Run the gray CheckDisplays test on flat panels
-check_fp_stripes	Run the stripe CheckDisplays test on flat panels
-check_hdmi <string>	Test an explicit HDMI mode. Pass -check_hdmi ? for details.
-check_hotplug	Run the HotPlug/Unplug test.
-check_kernel_ver <string>	Check whether MODS is running on the specified kernel
-check_linkspeed <string> <string>	Test if RM initialized the PexDev device and downstream device with the right speed
-check_linkwidth <string> <string> <string>	Test if RM initialized the PexDev device and downstream device with the right width
-check_pxl <string>	Test for explicit number of PCI-X lanes
-chipset_aspm <string>	Configure chipset ASPM setting.
-clock_slowdown <string>	Clock slowdown-global, nv, host, and thermal
-cml_training <string>	Enable/Disable types of CML training.
-cmos_training <string>	Enable/Disable types of CMOS training.
-compute	Run the compute manufacture tests.
-concurrent_devices	If true, will run MODS on multiple GPUs concurrently and synchronize test starts
-concurrent_devices_abort_on_error	Abort testing on all devices when one device fails during concurrent testing
-concurrent_devices_sync	If true, will run MODS on multiple GPUs concurrently and synchronize test starts
-corr_error_tol <string>	tolerance for number of correctable errors for each pcie link
-count_edc_error	Count EDC errors during mods tests
-csum_report	Report checksum differences for quals.

-cuda_in_sys	Put CUDA in system memory.
-cvb_check_ignore	Ignore the cvb check
-dd_scaler_mode <string>	Digital display scaler mode; native, scaled or centered.
-decode_error <string>	Print the test name and error description given a code.
-deep_idle_pstate <string>	Set the pstate to go into deep idle.
-def_powerrail_funcs <string> <string> <string>	Import and specify the Set/Get power rail functions
-detect_dfp	Detect a DFPs resolution to find goldens
-dev <string>	Select a resman device
-device_id <string>	Check the device ID
-disable_bc	Disable broadcast
-disable_def_img	Disable the default image on display.
-disable_dpu_sc_dma	Disables DPU SC DMA feature
-disable_edc	Disable EDC
-disable_elpg_on_init	Disable Elpg on Init for all engines
-disable_fail_message_print	Disable printing of FAIL message in big bold red block if mods fails
-disable_fandiag <string>	Do not initialize fan-diag
-disable_mods_console	Disable the MODS console
-disable_passfail_msg	Disable the big PASS/FAIL message at the end of mods.
-disable_pgob	Disable power gate on boot
-disable_pstate20	Tell RM to disable PState 2.0 on all boards.
-disp_ignore_imp	Set regkey for RM to disable IsModePossible functionality.

-dispclk <string>	Display clock in mhz.
-display <string>	Run the tests on the specified display.
-display_clones <string>	Combined mask of displays to enable clone mode on.
-display_config	Get the display configuration.
-display_manual_updates	Only send display updates explicitly requested by tests
-dramclk <string>	DRAM clock in MHz.
-dramclk_percent <string>	Set dramclk to X % of default. 50 <= X <= 150
-dump_png	Dump a .PNG file on Golden Store and Error events.
-dyn_eng_ctrl <string>	Enable (1 or 2) or disable (0) dynamic engine control.
-dynamic_mempool <string>	Enable or disable dynamic mempool. 0=disable, 1=enable
-early_exit_on_err_count	Exit tests early when error count is violated
-ecc_async_scrub <string>	ECC Asynchronous scrubbing: Enable = 1, Disable=0
-ecc_dbe_tol <string> <string>	tolerance for number of ECC double bit errors, expects <fb, l2, l1, sm> <tol>
-ecc_fuse_ignore	Ignore the ecc fuse
-ecc_sbe_tol <string> <string>	tolerance for number of ECC single bit errors, expects <fb, l2, l1, sm> <tol>
-ecc_verbose <string>	Bitmask for ECC verbose reporting (bit 0 = print on checkpoint, bit 1 = print on count change)
-edc_rate_limit <string>	tolerance for maximum of EDC error rate, Maximum value 0xFFFFFF
-edc_tol <string>	tolerance for overall number of EDC CRC errors
-edc_verbose <string>	Bitmask for EDC verbose reporting (bit 0 = print on checkpoint, bit 1 = print on count change)
-elcg <string>	Engine Level Clock Gating.
-elcg_off	Engine Level Clock Gating Off.

-elpg_idle_thresh <string> <string>	Set the PowerGate Idle threshold (in clocks) (usage : <gr|vid|vic|ms> clocks)
-elpg_mask <string>	Mask for enabling ELPG
-elpg_off	Engine Level Power Gating Off.
-elpg_ppu_thresh <string> <string>	Set the PowerGate Post Powerup threshold (in clocks) (usage : <gr|vid|vic> clocks)
-enable_aelpg	Enable AELPG by setting RM registry.
-enable_clk2 <string>	Enable Clocks 2.0 in Resman
-enable_ecc_inforom_reporting	RM to blacklist pages in InfoROM on ECC error
-enable_gen2	Allow RM to transition PEX speed to Gen2.
-enable_gen3	Allow RM to transition PEX speed to Gen3.
-enable_hda <string>	Enable or disable HDA engine. 0=disable, 1=enable
-enable_no_snoop	Enable No Snoop of the host/FIFO through registry-control
-enable_nvdps	Enable nvDPS (hardware detection of screen activity)
-enable_pgob_compute	Enable power gate on boot for compute
-enable_pgob_dfma	Enable power gate on boot for DFMA
-enable_pgob_tex	Enable power gate on boot for tex
-enable_replayable	Enable RM functionality for generating replayable log files
-enable_ticks	Enable ticks in the OpenGL driver
-end_dump_addr <string>	End BAR0 address to log.
-errcode_test_offset <string>	Add this to testnumber in error codes.
-etmp_range <string> <string>	Set min, max degrees Celsius for External temp sensor sanity-check.
-exit_on_breakpoint_count <string>	Exit MODS when the breakpoint count is reached (0 = don't abort)

-ext_banks <string>	Test for explicit number of external banks
-external_heap	Forces RM to use external heap mgmt (similar to Vista).
-extra_pexcheck	Check for CORR PCIE Errors inside supported GpuTests.
-fail_critical_fb_range <string> <string>	make FB errors in this range critical
-fan_speed <string>	Force current gpu devices fan to this % of max.
-fb_gddr5_x16	Enable GDDR5 x16 FB mode
-fb_gddr5_x8	Enable GDDR5 x8 FB mode
-fbi_check <percent>	Set dram overclock during retest to determine if a test failure is due to FB problems. Default 15, set to 0 to disable this feature.
-force <string>	Add and Force the specified test(s).
-force_SMbus_temp <string> <string> <string>	Force displaying / checking temperature with Smbus external sensor.
-force_coh	Force all surfaces and pushbuffers to coherent (cached sysmem).
-force_ecc_L1	Force enable ECC L1
-force_ecc_SM	Force enable ECC S
-force_ext_temp <string>	Force displaying / checking temperature with external sensor.
-force_fb	Force all surfaces and pushbuffers to FB memory.
-force_gl_coh	Force only GL tests to use coherent (cached sysmem).
-force_hdmi_info	Force sending HDMI info frames.
-force_head_routing <string>	Specify head indexes for active displays - 4 bits per head.
-force_int_temp <string>	Force displaying / checking temperature with internal sensor.
-force_ipmi_temp <string> <string> <string>	Force displaying / checking temperature with Ipmi external sensor.

-force_ncoh	Force all surfaces and pushbuffers to noncoherent (uncached sysmem).
-force_ncoh_sysmem	Force sysmem surfaces and pushbuffers to noncoherent (uncached sysmem).
-force_repost	Force repost of the GPU.
-force_small_pages	Forces the use of small pages while allocating framebuffer memory.
-force_vga_print	Print to VGA screen regardless of whether user interface is enabled or disabled
-foreign_display	Don't send any display commands to HW. Instead copy rendered output to a foreign display, not recognized by RM.
-foreign_display_dev <string>	Device index of device on which to set up foreign display (default is 0).
-foreign_display_fps <string>	Number of frames per second copied to foreign display.
-freq_offset_khz <string>	Shift the VF curve by y Khz.
-fspg <string>	Floorsweep Power Gating.
-fspg_off	Floorsweep Power Gating Off.
-full_gc5	GC5 instead of GC5 minus
-full_power	Full power mode
-full_power_display	Full power for display
-fullpower	Full power mode
-gl_force_mem_space	Force surfaces other than Z/color to SysMem only.
-gl_force_sysmem_buffers	Force render to SysMem only.
-gl_no_zbc	Disable GL zero-bandwidth-clear updates.
-gl_one_channel	Ask GL driver not to flush the GLS channel (reduces context switching).
-glkey <string> <string>	Set a registry key for OpenGL.

-global_surface_overrides <string>	Set the GlobalSurfaceOverrides registry key.
-glr_frame_retries <string>	Set the FrameRetries on all glrandom tests (used for reporting soft/hard)
-glrandom	Run the GLRandom tests
-glsbg <string>	Run GLStress in background on dev N
-goldenfile <string>	Specify the golden value file
-gpc2clk <string>	GPC2 clock in mhz.
-gpc2perf <string>	Sets Gpc2 clock (and related perf parameters in PState 2.x)
-gpc_mask <string>	Set GPC enable mask.
-gpcclk <string>	GPC clock in mhz.
-gpio_activity <string> <string> <string> <string>	enables GPIO activity monitor. User provides the edge to look for + threshold number
-gpu_aspm <string>	Configure GPU ASPM setting.
-gpu_cache_alloc_policy <string>	Set the GPU cache allocation policy
-gpu_cache_promotion_policy <string>	Set the GPU cache promotion policy
-gpu_cache_write_mode <string>	0=default 1=writeback 2=writethrough
-gpu_codec <string>	Azalia port number for gpu output
-gpu_dma_tm <string>	Run GpuDmaTest in a particular mode
-gpu_num <string>	Set which graphics processor to test (default 0).
-gpu_out <string>	Azalia codec index for gpu output
-grctx <string>	Set context switching mode. 1=hybrid, 2=hw, 3=sw
-grreginitoverride <string>	Set graphics register init override behavior. 1=prod diff
-h	Display Help

-hasbug_override <string> <string>	Override Has Bug.
-hd_codec_sdi <string>	SDI Line the GPU Codec is connected to for the HD Codec test
-hdcp_adksv_only	Only check HDCP A and D keys
-hdcp_delay <string>	Settle time for CheckHDCP test
-hdcp_keys	Get the HDCP keys. Not a complete test.
-hdcp_loops <string>	Settle time for CheckHDCP test
-hdcp_skip	Don't do HDCP for interactive display tests
-hdcp_timeout <string>	Timeout for negotiating an HDCP connection
-hdmi_fft_ratio <string>	Set signal energy ratio to determine pass/fail
-help	Display Help
-hostclk <string>	Host clock in MHz.
-hw_speedo_override <string>	Override hw speedo value
-id	Prompt user for test ID, which is written to the logfile.
-iddq_check_ignore	Ignore the iddq check
-idle_channels_retries <string>	Retry count when idling channels using polling idle (default: 0)
-idle_slowdown <string>	Sets override for IDLE slowdown settings.
-ignore_family <string>	Ignore a gpu family (curie, tesla, fermi, kepler) during MODS init
-ignore_fatal_errors	Ignore Fatal PEX errors
-ignore_gr_checksum	Do not use graphics checksum for identification
-ignore_ot_event	Ignore thermal overtemp events.
-ignore_unexpected_interrupts	Ignore unexpected gpu interrupts in tests. Please use with caution.

-inst_in_sys	Put instance memory in system memory.
-int_therm_calibrate <string> <string>	Thermal Calibration of Internal Sensor
-intr_thresh <string>	Set the stuck interrupt threshold for ResourceManager.
-ipmi_temp_range <string> <string>	Set min and max ipmi temperature range - trigger errors for tests.
-itmp_range <string> <string>	Set min, max degrees Celsius for Internal temp sensor sanity-check.
-json	Enable single-run JSON logfile to modsNNNN.log.
-json_append <string>	Enable multi-run JSON logfile, appending to file
-json_clobber <string>	Enable multi-run JSON logfile, overwriting file
-json_name <string>	Enable single-run JSON logfile, and specify filename template
-kickoff_thresh <string>	Set channel auto-flush threshold.
-l2_mode <string>	Sets the L2 Mode (1 = bypass)
-legacy_interrupts	Use legacy GPU interrupts.
-legacyclk <string>	Legacy clock in mhz.
-line_in <string>	Azalia port number of the LINE-IN audio jack
-line_in_codec <string>	Azalia codec index for LINE-IN
-link_speed_override <string>	Override link speed of a perf point
-link_width_override <string>	Override link width of a perf point
-list_errors	List all the MODS errors.
-list_tests	List all the MODS tests and their test numbers.
-log_file_limit_mb <string>	Limit of the log file size in Mb.
-log_imp	Capture IsModePossible log

-log_imp_io	Dump IsModePossible I/O
-logcmp	Dump logcmps.
-loops <string>	Loop the tests count times.
-low_power	Lower Power Settings
-lowest_power	Lowest Power Settings
-lowestpower	Lowest Power Settings
-lowpower	Lower Power Settings
-ltc2clk <string>	LTC2 clock in mhz.
-ltcclk <string>	LTC clock in mhz.
-lvds_loop_clk <string>	Pixel Clock frequency in LVDS loopback test.
-mats_cov <string>	Override Mats memory coverage percentage (0 to 100), default is 10.
-mats_rd_delay <string>	Delay in NS before a fb Read
-mats_wr_delay <string>	Delay in NS before a fb Write
-matsinfo	If a mats-derived test fails, print out more info
-max_ext_minus_int <string>	Set max diff of (external - internal) temperature value, in degrees C (default 7)
-max_int_minus_ext <string>	Set max diff of (internal - external) temperature value, in degrees C (default 7)
-max_pwr_range <string> <string> <string>	Set the maximum power range on <sensor> <min> <max>
-max_temp_diff <string>	Set max internal vs. external temperature mismatch, in degrees C (default 7)
-maxframes <string>	Limit max frames per test (shorten test times).
-maxmemerr <string>	If a mats-derived test fails, print at most that many errors.
-maxwh <string> <string>	Set max screen resolution (w, h only).

-mboard <string>	Enable multiboard (SLI) modem with device N as master
-memqual	Enable memqual specific (RM) behavior.
-message_disable <string>	Disable debug messages from mods modules. Separate with colons, e.g. -message_disable ModsCore:ModsNvGpu
-message_enable <string>	Enable debug messages from mods modules. Separate with colons, e.g. -message_disable ModsCore:ModsNvGpu
-mfg	Run the board manufacturing tests.
-mfg2	Run the board manufacturing tests.
-min_fb_mem_percent <string>	Minimum fraction of framebuffer memory that needs to be allocated
-min_mempool <string>	Enable or disable minimum mempool. 0=disable, 1=enable
-mobile	Tell RM that this is a Mobile GPU
-mode <string> <string> <string> <string>	Set mode X x Y @ Z bpp at W Hz.
-monitor <string>	Monitor the GPU status every X milliseconds.
-mpeg_in_fb	Run MPEG tests out of framebuffer rather than AGP.
-msdclk <string>	MSD clock in mhz.
-msi_interrupts	Use MSI protocol for GPU interrupts.
-multiheap_en	Enable MODS multi-heap code
-must_be_supported	Treat unsupported tests as errors
-no_autoflush	Disable auto-flush of channels.
-no_backdoor	Disable FB backdoor.
-no_compress	Disable FB compression.
-no_dynamic_mempool	Indicate to the RM to not use the dynamic mempool.

-no_ecc_fb_scrub	Don't scrub FB
-no_ext_power	Do not check if external power is connected.
-no_fs_restore	Do not restore floorsweeping state on GPU Subdevice shutdown
-no_gart	Disable NVGART.
-no_gen2	Disallow RM to transition PEX speed to Gen2.
-no_gen3	Disallow RM to transition PEX speed to Gen3.
-no_glext <string>	Disable one feature at a time
-no_gold	Do not load golden values.
-no_golden_dma	Use CPU reads for surface readbacks/CRC.
-no_inst_in_sys	Keep instance memory in FB.
-no_mseq	Do not use the sequencer when changing the memory clock
-no_pex_aspm	Disable PCI-E ASPM.
-no_pstate_lock_at_init	Do not lock to a pstate at initialization
-no_pte_in_sys	Keep page table entries in FB.
-no_rc	Disable robust channels.
-no_rcwd	Disable robust channels watchdog.
-no_require_fos	Allow glr_display to pass even if we can't FOS because we are on head 1.
-no_restore_aspm	Dont restore ASPM on exit.
-no_restore_clocks	Dont restore clocks on exit.
-no_shader_cache	Disable GL binary shader cache.
-no_sim_symbols	Do not load sim symbols when dumping sim stack

-no_sse_memcpy	Disallow SSE 16-byte reads in memcpy.
-no_stack_dump	Do not dump call stack
-no_temp_range	Disables temperature range check after each gputest
-no_thermal_slowdown	Disable thermal slowdown.
-no_twod_in_wfmats	Don't use the TwoD class if WfMats, (default for >= G80)
-no_vga	Disallow VGA text mode.
-no_wc_linux	Disable write-combining.
-no_zcull	Disable Zcull.
-non_coherent	Force all tests except Class1774 and Class3174 to use NonCoherent memory.
-nonconcurrent_test <string>	Run this test one device at a time with -concurrent_devices.
-notest	Get ready to run tests, but don't actually run them.
-notiled	Do not use tiled surfaces.
-null_display	Don't send any display commands to HW.
-nvlink_disable_clock_init	Disable NvLink clock initialization
-nvlink_disable_error_rate_check	Disable NvLink error checking by rate
-nvlink_force_config	Forcely configure NvLink
-nvlink_force_disable	Force disable NvLink
-nvlink_force_enable	Force enable NvLink
-nvlink_verbose	Set NvLink verbose mask
-nvlink_skip_check	Skips checking NvLink error counters
-nvlink_rx_crc_flit_tol	NvLink Rx CRC flit error tolerance (count/min using rate, count for non-rate)

-nvlink_rx_lane_crc_tol	NvLink Rx per-lane CRC error tolerance (count/min using rate, count for non-rate)
-nvlink_tx_replay_tol	NvLink Tx Replay tolerance (count/min using rate, count for non-rate)
-nvlink_tx_recovery_tol	NvLink Tx Recovery tolerance (count/min using rate, count for non-rate)
-num_fifos	Override the number of FIFOs RM is allowed to allocate
-oceb_size <string>	Override OCEB Size
-old_gold	Using old golden values.
-only_family <string>	Ignore gpus not in this family (curie, tesla, fermi, kepler) during mods init
-only_pci_dev <string>	Make RM see only the specified PCI device (format is bus:dev.func - all in hex)
-opsb_override <string>	set RM regkey to override the Optional Power Saving Bundle fuse/vbios values
-oqa	Run the outgoing quality assurance tests.
-outputfilename <string>	Temporary hack to WAR the fact that testgen always passes outputfilename
-override <string>	Execute the script file to override GPU settings.
-override_fb_size <string>	New FB Size in MB.
-pclk_overclock_pct <string>	Set display PClk overclock percent
-perlink_aspm <string> <string> <string>	Set ASPM for each PEX device. Parameter is Depth, Loc ASPM, Host ASPM
-perlink_corr_error <string> <string> <string>	Set the allowed number of CORR error per PCIE node. Parameter is Depth, LocTolerance, HostTolerance
-pex_crc_tol <string>	Tolerance for number of PEX Crc errors
-pex_l0s_tol <string>	Tolerance for number of PEX L0s Failed exits
-pex_line_error_tol <string>	Tolerance for number of PEX line errors
-pex_nak_rcvd_tol <string>	Tolerance for number of PEX NAK Recieved errors

-pex_nak_sent_tol <string>	Tolerance for number of PEX NAK Sent errors
-pex_non_fatal_rate_tol	Tolerance for number of non fatal PEX rate
-pex_verbose <string>	Enable verbose PEX reports
-pg_offload <string>	Override the PG log operational parameters
-pg_pr_gate_enable <string> <string>	Set the PowerGate Power Rail Gate enable (usage : index <0|1>)
-pg_pr_idle_thresh <string> <string>	Set the PowerGate Power Rail Idle threshold (in clocks) (usage : index clocks)
-pg_pr_predictive_thresh <string> <string>	Set the PowerGate Power Rail Predictive threshold (in clocks) (usage : index clocks)
-pgctrl_abort_timeout <string>	Override the abort timeout value used to abort PG ON process.
-pgctrl_parameters <string>	Override various PG Controllers settings.
-pglog_parameters <string>	Override the PG log operational parameters
-pglog_surface <string>	Override the PG log surface attributes
-pgob_mask <string>	Set bitmask for power gate on boot
-pll_settle_time <string>	PLL settle time in nanoseconds
-pmu_bootstrap_mode <string>	PMU Bootstrap Mode
-pmu_force_phys <string>	Force all PMU memory mappings to be physical
-pmu_instloc_inst <string>	Location of the PMU instance block (def/coh/ncoh/vid)
-pmu_instloc_ucode <string>	Location of the PMU ucode surface (def/coh/ncoh/vid)
-pmu_ucode_addrmode <string>	PMU Ucode Addressing Mode
-poll_hw_hz <string>	Max frequency at which to poll hw registers.
-poll_interrupts	Poll for GPU interrupts.
-power_cap_max	Set all power limits to the max allowed in vbios

-power_cap_policy <string> <string>	(policyIdx, mw) Set a power-capping policys mW or mA limit.
-power_cap_rtp <string>	(mw) Set Room Temperature Power (soft limit)
-power_cap_tgp <string>	(mw) Set Total GPU Power (hard limit)
-power_feature <string>	Set power feature
-power_feature2 <string>	Set power feature
-preheat <string> <string>	Preheat the chip to a certain temperature.
-print_is_supported	List IsSupported responses from all tests in the current spec.
-print_sys_time	Print [hh:mm:ss] prefix on all messages.
-print_tests_to_run	Simply prints the tests that would be run in the current configuration.
-print_wallclock_time	Print [hh:mm:ss] prefix on all messages.
-printcsv	Enable Golden.PrintCsv mode.
-privsec_disable	Disable Priv Security for debugging purpose
-pstate <string>	Test only this pstate.
-pstate_disable	Do not allow RM to change clocks or initialize perf tables
-pstate_hard	Use "hard" pstate locks on all devices
-pstate_soft	Use "soft" pstate locks on all devices
-pte_random	Allocate all system memory in randomly scattered pages.
-pte_reverse	Allocate all system memory from top down.
-pte_seed <string>	Random seed for scrambling system memory allocations.
-pwr_cap <string>	1:enable, 0:disable SmartPower power capping
-pwr_rail_gate_off	Disable power rail gating

-pwr_rail_gate_on	Enable power rail gating - and ELPG as well
-pwr_range_mw <string> <string> <string>	Set min and max power range for a sensor - trigger errors for tests.
-pwrclk <string>	Power clock in mhz.
-queued_print_enable <string>	Enable or disabled queued printing (multithread support), 0 = disable, 1 = enable
-ram_config_strap <string>	Kind of FB memory attached to Gpu.
-random_prompt	Randomize input for check_displays correct image prompt
-rc_timeout_sec <string>	Set timeout value for Robust Channels in seconds.
-readspec <string>	Use the user defined spec.
-reg_write_mask <string>	Store register ranges for GpuInstances alongwith bitmask to be used to write them
-regress_using_gltrace	Run regressed loop using gltrace
-regwr <string> <string>	Override gpu register (addr, value)
-regwr_early <string>	pre VBIOS register setting
-regwr_mask <string> <string> <string>	Override gpu register (addr, andMask, orMask) - do AND first then OR
-relocate_inst	Forces instance space at the begining of FB
-require_displays <string> <string>	Specify required present and not-present display masks.
-reset_exceptions_on_exit	Reset exceptions when shutting down gpu subdev.
-restore_clocks	Do restore clocks on exit.
-retry_copy_check_on_fail <string>	Retry Comparision in system memory if any miscompare found
-reverse_lvds_tmds	Reverse LVDS and TMDS entries in VBIOS
-revision <string>	Check the NVIDIA chip revision.
-rm_clients <string>	Number of RM clients to create

-rmmsg <string>	Override RM Message handling.
-rmsbg <string>	Select which device to run the RMStress Background test on.
-rom <string>	Check the rom version.
-run_after_init <string>	Execute specified javascript after GPU initialize.
-run_on_error	Continue running if error occurs.
-run_only_gold	Only run tests that use golden values.
-run_only_hw_crc	Only run tests that use DacCrc, TmdsCrc, and/or TvCrc.
-runlist	Enable buffer-based runlists
-runlistsize <string>	Set the size of buffer-based runlists. Must be a power of 2.
-safe_dmas	Use safe DMA protocol rather than fast.
-savespec <string>	Save specified spec to file
-savespec_args <string>	Save string of arguments in user defined spec.
-screen_off	Disable output to the screen during tests.
-seed <string>	Random number seed.
-serial_ports <string>	Total number of serial ports to be tested.
-set_canoas_mapping <string> <string>	Set the mapping of a GpuInstance to SysconId + GpuIndex (inside a syscon).
-set_nvvdd_of_pstate <string>	Set NvVdd based on PState number.
-set_power_cap <string>	Set Canoas Power Cap in Watt. 0=disable.
-set_powerrail_leakage_thresh <string> <string>	Set the power rail thresholds
-set_powerrail_voltage <string>	Set the power rail voltage at init

-setgpio <string> <string>	Toggle GPIO after RM initializes.
-show_gold	Display contents of goldenXX.bin file.
-sim_int_temp <string>	Simulate internal thermal sensor temperature. Requires security license.
-simulate_all_dfps <string>	Simulate a flat panel with the specified EDID on all possible DFPs.
-simulate_dfp <string>	Simulate a flat panel with the specified EDID.
-skip <string>	Skip the specified test(s).
-skip_board_detect	Skip board detection in the ValidSkuCheck test.
-skip_config_sim	Skip simulation configuration during initialization
-skip_fan_rpm_sense	Skip fan RPM sense portion of CheckFanSanity.
-skip_fan_sense	Skip fan sense portion of CheckFanSanity.
-skip_inta_intr_check	Skip the INTA interrupt check done at Gpu initialization.
-skip_msi_intr_check	Skip the MSI interrupt check done at Gpu initialization.
-skip_pertest_pex_speed_check	Skip the init and per GPU test PEX speed check
-skip_pertest_pex_width_check	Skip the init and per GPU test PEX width check
-skip_pertest_pexcheck	Skip the init and per GPU test PEX check
-skip_rm_state_init	Skip RM init; just do the VBIOS init
-skip_upstream_check	Make PEX tests check only the link directly above GPU
-slcg <string>	Second Level Clock Gating.
-slcg_off	Second Level Clock Gating Off.
-sli	Enable multiboard (SLI) mode, with the primary device as master.

-sli_always_approved	Force SLI approval in RM
-sli_config <string>	Bitmask of GPUs to be linked into an SLI configuration
-sli_master <string>	Enable multiboard (SLI) mode, with device N as master.
-sli_only_dir <string>	Force testing of SLI connector only in one direction
-sli_pixel_clock <string>	Select Pixel Clock in MHz for SLI testing
-sli_use_display <string>	Select display output connector for SLI testing
-slt	Run the chip manufacturing tests.
-smbus_temp_range <string> <string>	Set min and max smbus temperature range - trigger errors for tests.
-soak <string>	Soak the chip for given seconds.
-sockCmdLine <string>	Pass Command to start sockserver.
-sor_loadadj <string>	Set PLL1 LoadAdj value in SOR loopback test.
-sor_pattern <string>	Run only the selected pattern in SOR loopback test.
-spdif_codec <string>	Azalia codec index for SPDIF output
-spdif_out <string>	Azalia port number of the SPDIF-OUT audio jack
-spec <string>	Use the user specified table for this mode.
-strap_fb <string>	Set the framebuffer strap in megabytes.
-subdev <string>	Select a resman subdevice.
-subsystem <string> <string>	Check the subsystem vendor and device IDs.
-suggest_pstate_at_init	Suggest a PState at init
-swSlowdown	Enable clock slowdown
-swap_endian	Run GPU in big endian mode on a little endian computer.

-sys2clk <string>	Sys2 clock in mhz.
-sysclk <string>	Sys clock in mhz.
-syspll <string>	Sys PLL clock in mhz.
-tc_weak_key	Use weak turbo cipher key
-tco	Allow Nforce systems to reboot automatically in a crash
-temp <string>	Control gpu fan to reach given temperature (if tgt < 0, just report temps).
-test <string>	Run only the specified test(s).
-test_for_gen1	Tell ValidSkuCheck that were are intentionally testing with Gen1 chipset.
-test_gpu <string>	Name of the gpu to be tested
-testarg <string> <string> <string>	(test, property, expression) sets tests property to result of expression.
-testargstr <string> <string> <string>	(test, property, str) sets tests property to str
-testforce <string>	Run only and force the specified test(s).
-testhelp <string>	Display available -testarg options for a given test number.
-threadid	Turn on the prepending of the thread ID to a line of log spew
-time	Record duration of tests.
-timeout_ms <string>	Set default Timeout in MS (for both Tests and RM).
-tmds_crc	Use TMDS/LVDS CRCs.
-tmds_loop_clk <string>	Pixel Clock frequency in TMDS loopback test.
-tmds_pllreg <string>	Set PLL0 PllRegLevel value in TMDS loopback test.
-tmds_txreg <string>	Set PLL0 TxRegLevel value in TMDS loopback test.
-tpc_mask <string>	Set TPC enable mask.

-tpc_mask_on_gpc <string> <string>	Set TPC enable mask for the given GPC. (usage: <gpc_num> <mask>)
-trepfile <string>	Set the name of the trep (test report) file
-un_supp_req_tol <string>	Tolerance for number of unsupported request for each pcie link
-unlock_aslm	Unlock ASLM on the current chipset
-unlock_chipset_gen2	Unlock Gen2 capability on chipset that RM prevents from going into Gen2.
-use_dynamic_mempool	Indicate to the RM to use the dynamic mempool.
-use_mods_console	Use the MODS console
-use_orig_fb_req_size	Indicate to the RM to use the original (unpadded) size of the FB alloc request for comptag calculations.
-use_perfpoints <string>	Add a PerfPoint for each pstate to test: min/nom/mid/tdp/etc
-use_raw_console	Use the raw console
-use_rc_callback	Use an RM callback for robust-channel errors (aka two-stage recovery).
-use_vfpoints	Build PerfPoint for each vfpoint
-use_virtual_dma	Use GPU virtual addressing for accessing memory.
-user_strap <string>	Check user strap value in CheckConfig.
-va_reverse	Allocate virtual addresses from the top down
-vas_ram_size <string>	Set Ram size to scale VAS size (option: 0, 1 or 2)
-verbose	Run the tests in verbose mode.
-verify_fuse <string> <string>	Append fuse & spec to the list to be checked for Test 1 - CheckConfig
-verify_lanes <string>	Fail if PCIE lanes are different than specified number.
-verify_sku <string>	Checks if the fuses are burnt for the given sku

-verify_suspect_shader <string>	Pass in a special js filename to verify a suspect shader
-volt_offset_mv <string>	Shift the VF curve by x mV.
-volterra_max_temp_delta <string>	Set the max difference between two consecutive readings on the same volterra slave (default = 15)
-volterra_temp_range <string> <string>	Set min and max volterra temperature range - trigger errors for tests.
-weak_skey	Disable DH Key Exchange
-wlm_enable <string> <string>	Enable WLM during test with optional freezeUsec runUsec Ratio(-wlm_enable x y).
-xbar2clk <string>	XBar2 clock in mhz.
-xbarclk <string>	XBar clock in mhz.
-zcull_mask_on_gpc <string> <string>	Set ZCULL enable mask for the given GPC. (usage: <gpc_num> <mask>)

3.5 Test Selection

MODS has an object-oriented test selection mechanism. Embedded in each test is a function that reports back whether that particular test is supported on the unit being tested. Therefore, normal MODS operation only requires that the user runs mods gputest.js –mfg (or –oqa, see section 2.0 above). When this default command-line is used, each test is queried to determine if it can run (i.e., the hardware and operating system both support it) and should be run (i.e., it is part of the selected test suite) on the unit being tested, and then this subset is run sequentially.

The following command-line options can be used to change this behavior:

Table 4. Test selection arguments

Option	Description
-add X	In addition to running the normal suite of tests, run test X if it can be run, regardless of whether it should be run. If test X can't be run, ignore the request and silently not run test X. To force

	execution of test X, use the –force option.
-skip X	Run the normal suite of tests, but skip test X. Using –skip on the same test number as –force or –add will cause an error. However, combinations of these command-line options are legal if they do not refer to the same test number.
-test X	Do not query each individual test to determine if it should be run. Run only test X. –test and –skip cannot be used simultaneously.
-force X	Run test X even if it can't be run and/or shouldn't be run. This may result in errors if the hardware being tested does not support test X. This option attempts to run test X in addition to the normal test suite.
-testforce X	Run test X even if it can't be run and/or shouldn't be run. This may result in errors if the hardware being tested does not support test X. This option runs only test X.

Using –test on the same invocation with –add, -force or –skip will cause an error, even if they refer to different tests.

3.6 Installation

Place all distribution package files into a single directory.

On MacOSX, click on the ".tgz" package to unpack it. To run MODS, type "./mods gputest.js" or another command line in the "Mods.app/Contents/Resources" directory. Alternately, you can edit "Mods.app/Contents/Resources/mods.arg" to contain the command line you want to run, then click on the MODS icon

3.7 Prerequisites for Running Linux

This section applies to users who wish to run MODS on a Linux distribution other than the one provided by NVIDIA described in section 3.9. If you are using the NVIDIA-supplied distribution you don't need to read this section.

Linux manufacturing MODS requires minimum kernel version of 2.6.18. Version 2.6.29 or newer is recommended for performance reasons. Older versions have not been tested and may not be working. Kernel 2.4 is not supported. The version of the running kernel can be established by running:

```
$ uname -r
```

The 'check_config.sh' script can be used to ensure that the Linux system has sufficient requirement.

Linux manufacturing MODS is a 64-bit application and it requires kernel compiled for x86_64 architecture. To determine kernel architecture, type:

```
$ uname -m
```

The system on which MODS is run must be built on glibc-2.5 or newer. To determine glibc version, type:

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

Linux manufacturing MODS includes a kernel module. The purpose of this module is to expose certain kernel-mode APIs to MODS, 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's name is kernel-sources or linux-sources. For example on Debian, type:

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

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

For successful MODS runs the NVIDIA GPUs in the system must be in their original, unaltered state, as initialized by VBIOS. This means X must not have been run on the NVIDIA GPUs prior to running MODS. Please 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).

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 MODS to function correctly, this driver has to be unloaded, preferably blacklisted so that it is not automatically loaded at boot.

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=.

3.8 Installing the Kernel Module

This section applies to users who wish to run MODS on a Linux distribution other than the one provided by NVIDIA described in section 3.9. If you are using the NVIDIA-supplied distribution you don't need to read this section.

Linux MODS relies on a kernel driver to handle cases where it is necessary to use kernel-mode APIs. The easiest way to install the MODS kernel module is to use the provided installation script, which you will find in MODS 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 an 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 it's the video group, like for the nvidia driver. Please 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 what 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:

```
$ 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.

3.9 Creating a Linux Disk Image

NVIDIA distributes a turnkey Linux package that can be used to create a disk image. This package can be obtained on the internal NVIDIA network. Customers should contact their NVIDIA representative to obtain this file. This distribution will fit on a 64 MB flash drive.

The package default.zip can be installed on a target drive (e.g. USB stick) from Windows XP. The installation procedure is as follows:

- Insert the USB stick or make sure the drive where you want to install it is connected.
- Format it using FAT32 filesystem. For USB sticks right click on the drive and choose "Format...". For non-removable drives (such as SATA) you need to go to Control Panel->Administrative Tools->Computer Management->Storage->Disk Management and create a partition smaller than 32GB (max size for FAT32) and then format it.
- Unzip the modsdisk.zip package to the target drive you've just formatted e.g. by right clicking on the zip file in Explorer, choosing "Extract all..." and entering drive letter (e.g. e:) as the destination.
- Open command prompt (e.g. Start->Run... and type "cmd<ENTER>").
- Go to the target drive by typing drive letter of the target drive and pressing enter (e.g. "e:<ENTER>").
- Install the boot loader (assuming e: is your drive) (for non-removable drives you might need to also add the -f switch):
  - $ syslinux -m -a e:

3.9.2 Disk contents of the Linux Disk Image

File	Description

/ldlinux.sys	File used by the bootloader to load Linux.
/syslinux.exe	Bootloader installer for Windows.
/syslinux/kernel	Compressed Linux kernel image
/syslinux/initrd	Compressed initial ramdisk containing files needed to access the compressed Linux filesystem.
/syslinux/squash.bin	Compressed Linux filesystem with all files needed to run MODS.
/syslinux/syslinux.cfg	Bootloader configuration file containing command line for the Linux kernel.
/syslinux/commands	Linux shell commands which are executed after the system finishes loading itself. This file can be edited on Windows.
/mods/mods.tgz	MODS package.
/mods/args	Arguments for MODS. This file is explicitly specified in /syslinux/commands. It can be edited on Windows.

3.9.3 Usage

After you made the disk bootable, insert it or connect it and order BIOS to boot from it.

You can edit and customize the /syslinux/commands file to load additional kernel modules, initialize networking, send mods.log created by MODS over the network, etc.

You can also customize the MODS arguments in file /mods/args.

When you boot the Linux distribution it will load Linux and execute commands in /syslinux/commands which by default will launch MODS with arguments specified in /mods/args.

3.9.4 Running MODS again

The default image will run MODS immediately after boot. To run MODS again:

```
$ cd /mnt/nv/mods

$ /tmp/mods/mods @args
```

You can also replace "@args" with actual arguments.

3.9.5 How It Works

This is an explanation what happens from the moment BIOS boots from the Linux disk:

First the bootloader locates the /syslinux/syslinux.cfg file and finds where to find the kernel and the initial ramdisk.

The bootloader loads the kernel (/syslinux/kernel) and the initial ramdisk (/syslinux/initrd) to memory and uncompresses them.

Then the bootloader invokes the kernel code.

The kernel boots and initializes all devices it has drivers for.

After the kernel finishes booting it runs the /linuxrc script located in the initial ramdisk. This is our mods-linuxrc0 script.

The script mounts basic filesystems (/dev, /proc, /sys), finds the drive where the DOS filesystem is located, mounts it and then mounts the squashed Linux root filesystem. It also prepares a new ramdisk which will become the new, final root filesystem.

The script executes another, final /linuxrc script from the squashed root filesystem. This is our mods-linuxrc1 script. At the same time it rotates the root directory so that the final squashed root filesystem becomes the main root filesystem.

The second linuxrc script again mounts basic filesystems under the new root filesystem, loads MODS driver and then executes commands from /syslinux/commands.

4.0 GPU TESTS

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. Optionally, capture image file(s) in .TGA format for failure analysis.

Restore previous display mode and refresh rate.

Release screen to windowing 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.

4.1 Test Descriptions

Table 5. List of GPU tests

T#	Test Name	Description
1	CheckConfig	This configuration test is run if one of the command-line options listed below is used with gputest.js. -subsystem — Check the subsystem vendor and device ID's. -device_id — Check the device ID -revision — Check the NVIDIA chip revision. -foundry — Check the NVIDIA chip foundry. -require_displays — Specify required present and not-present display masks. -ram_config_strap — Kind of FB memory attached to Gpu. -user_strap — Check user strap value in CheckConfig. -verify_sku — Checks if the fuses are burnt for the given sku
2	GLStress	GLStress is an OpenGL-based graphics stress test. It first clears its surface to black, then repeatedly draws a variety of textured, lit triangle meshes that fill the surface. By using color logic-op

		XOR on an originally black surface, and always drawing each mesh twice, we know that the result should be black. Any rendering inconsistency will result in errors that will be preserved to the end of the test by the subsequent XOR drawing. At the end of the test, we report a failure if any pixels are not black. In addition to lighting and texturing, the test uses depth-test and stencil-test. This test is not included in some MODS builds.
3	MatsTest	A generic frame buffer memory test designed to catch coupling faults within memory arrays. Stepping both up and down through the array as well as alternating reads and writes is important for catching certain cases of array-coupling faults. The test indicates which data bits fail, front or back banks, which memory lane, and whether the failure was on a read or a write. The –matsinfo flag can be used in conjunction with this test for extra error reporting.
4	EvoCurs	Test the cursor rendering circuitry. This test randomly positions the cursor and performs a DAC CRC to verify if the rendered cursor is correct. This test cycles through all combinations of display devices so that all heads get tested.
7	EvoOvrl	Test the GPU's overlay video circuitry. This test reads a given YUV image from specific location with certain size, and renders it as an RGB image at a specific screen location, pixel size, and magnification. A DAC CRC is used to verify if the rendered image is correct.
8	CheckHotPlug	Interactive display hotplug test.
9	Random2d	This is the same as Rnd2dTest.FbRun, but it renders to NonCoherent memory and is shorter. Both tests draw the exact same thing (in fact, the same CRCs are used).
11	EvoSor	Display pipeline loopback test for G80 and higher
12	EvoSli	Tests whether the SLI video-bridge is working correctly or not by computing CRCs of scanned-out image data.
17	ValidSkuCheck	The purpose of this test is to confirm that it matches a valid sku configuration. It is used to catch these failures:

		1. Cards that have been mis-sized by the video BIOS' sizing algorithm. 2. Incorrect fusing (including GPC/TPC/FBP/ROP&L2) 3. Board straps, BIOS settings (GL ness, ECC, PCIE config)
18	ByteTest	Just like Mats, except perform 8 bit reads/writes instead of 32 bit read/writes.
19	FastMatsTest	Similar to Mats, except use GPU hardware writes instead of CPU writes. The algorithm works like this: 1. It first divides the framebuffer into "boxes" of 64x64 pixels. 2. Render a pattern to a random rectangle 3. DMA the contents of that rendered box to system memory 4. Pick a different random rectangle and render a pattern to it (note: #3 and #4 occur in parallel to make the test more stressful) 5. Start the rendering of the pattern to the next random rectangle. 6. Check the contents of the box DMA'd to system memory in #3. (note: #5 and #6 occur in parallel to make the test more stressful) 7. Repeat steps 3-6 for all boxes 8. Repeat steps 3-7 for all patterns
22	CheckDisplay	Display a red-green-blue-white diagonal image on the specified display head and display type for a visual inspection. This is an interactive test.
23	GLStressDots	This OpenGL-based test renders offscreen and uses a GPU shader to check for errors without reading back the color buffer over the PCIE bus once per second. Dots are drawn to the

		screen to indicate loop count. It produces a slightly "bursty" power load (less total power than test 2) but at fixed frequency (varies per board).
24	CheckHDCP	Checks to make sure HDCP is working. Requires an HDCP-capable GPU and an HDCP-capable display. This test is enabled by default on GPUs that support HDCP. This means that if your card supports HDCP but you do not test with an HDCP display, the test will fail. This is the intended behavior. If the user wishes to waive HDCP testing, he or she can use the "-skip 24" command-line option.
25	MSDECTest	This is a test for the video decompression engine. This engine is also called "VP3."
28	NvlbwsBeHammer	Variant of NvLinkBwStress (246). Uses line access from block linear surface to pitch linear surface to trigger numerous byte enables.
30	PcieSpeedChange	The primary purpose of PcieSpeedChange is to test the ability to change between PCIE Gen1, Gen2, and Gen3 speed. There are two phases to this test: The first section switches the bus speed between Gen1, Gen2, and Gen3, and checks if we can read back the current speed correctly. The second part of the test switches bus speed, and does two DMA operations between system memory and frame buffer after the speed change. At the end of each DMA operation, MODS will also check whether a PCIE error has been flagged. The DMA operations will construct a 'result surface' that MODS will validate.
31	CheckThermalSanity	If there is a thermal measuring device on-board, this test makes sure that the values returned are reasonable and not out of bounds.
33	GetDisplayConfig	Get the display configuration, i.e. print the attached display devices on each display head.
34	CheckDisplayBars	Display red-green-blue-white bars on all display heads for a visual inspection. This is an interactive test.
35	CheckDisplays	Display a red-green-blue-white diagonal image on all display

		heads for a visual inspection. This is an interactive test.
36	StereoTest	Checks if the stereo hardware is working correctly. This is an interactive test and requires special hardware such as stereo glasses or an LED wired to the stereo pin output.
37	NvlbwsBgPulse	Variant of NvLinkBwStress (246). While the bandwidth tests runs, the test also kicks off CudaLinpackPulse in the bg to generate power supply noise.
38	CheckFpGray	Displays a special gray image on all flat panels for a visual inspection. This is an interactive test.
41	MultiBoardDma	This test is based on DmaTest.RunTest. It has been modified to keep 2 GPUs and the CPU busy at once to stress multi-board, including peer-to-peer DMA and broadcast.
42	CheckDisplayBar	Display red-green-blue-white bars on the specified display head and display type for a visual inspection. This is an interactive test.
43	SpdifCheck	A SPDIF cable check. The support chipset will output SPDIF signal out of the motherboard. An SPDIF cable coming out of motherboard should be plugged into the graphics card. Using the Azalia chipset, this test will output 3 different sampling frequencies and expect the GPU to see that the sampling frequency changed.
44	SecTest	GPU mfg test for the SEC (SECurity) engine. The SEC engine is a DMA engine that also handles encryption and decryption, as required by HD-DVD video data in memory-spaces that might be accessible to hackers trying to make unlicensed copies of movies. This tests a randomized sequence of transfers using all possible features. Encrypts and decrypts are checked by first confirming that the data mismatches the source data, then performing the inverse operation to restore the original data and checking again for a match.
45	CheckFpStripes	Display a special stripe image on all flat panels for a visual inspection. This is an interactive test.
46	DisplayCrcChecke	Automatically validate display CRC post connectors at various

	r	display modes. This requires a E1850 CRC checker device.
48	CheckFbCalib	The test checks if auto-calibration of FB interface ended with reasonable values. When auto-calibration ends with maximum strength values (which should be signaled by the test) it may mean that something is wrong with the interface and memory transactions to FB may become corrupted. Usually this can be corrected by changing voltages, FB calibration resistors on PCB, etc.
50	I2CTest	Check if the GPU's external I2C bus is properly equipped with pull-up resistors.
52	MarchTest	This is an alternate way to call the Mats test (see below). This version does a "marching ones and zeros" memory pattern.
54	GLRandomCtxSw	glr_ctxsw is glr_hwtest with GLStress running in the background. Both tests must pass.
58	Random2d	This is a combined 2d rendering test. It tests blit (rectangular pixel region copy), 2d line, triangle, rectangle, and text drawing, texture downloading and format conversion including palette lookup and dithering, image scaling and stretching, and video colorspace conversion with compositing.
63	Optimus	Tests the GPU power down, power up, and re-initialization in Optimus notebooks
65	CheckOvertemp	Checks if the GPU has overheated during the test.
69	CheckHiResCrcs	Check that the DAC can handle high-resolution video modes and still scan them out correctly.
70	PatternTest	This is an alternate way to call the Mats test. This version uses a memory pattern supplied by Apple.
71	AppleGL	A port of Apple's OpenGL test. This test only runs on the Macintosh version of MODS.
73	HeatStressTest	Thermal.RunStress is similar to RmStress.Run in that it runs the RM's internal stress test, but instead of using mods code for surface alloc/init, looping, and surface testing, it just does the

		same ConfigSet call that the win32 "auto-overclock" control-panel utility uses to see if a given nv/dramclk setting is stable. It doesn't return much data.
74	AppleAddrTest	This is a variant of Mats using an address-on-data pattern.
75	AppleKHTest	This is a variant of Mats using the Knaizuk Hartmann test pattern.
76	AppleMOD3Test	This is a variant of Mats using the MOD3 test pattern.
78	CheckFanSanity	Checks if: 1. If the fan is spinning or not 2. If the fan RPM at 100% PWM is at least 30% more than the RPM at 30% PWM. 3. If the fan RPM at 65% PWM is the average of the fan speeds at 30% PWM and 100% PWM with a 30% tolerance. 4. If the fan RPM at 30% PWM is more than 500 RPM. 5. If the fan RPM at 100% PWM is more than 2000 RPM.
79	TurboCipher	Test the TurboCipher data encryption engine.
81	GLRandomHw TestCrc	Obsolete Test. Replaced by new GLRandom tests. These are test 130 through 141.
83	CheckVbridge	Tests for the presence of an SLI video bridge. This test is enabled with –check_vbridge command-line option.
84	HdmiLoopback	Internal HDMI loopback test. HdmiLoopback test works in different ways depending on whether the gpu being tested has an onboard azalia controller or not. For GPUs that don't have on-board audio controller: The loopback for this test would be as follows: a. From SPDIF-out of chipset to SPDIF-in of GPU b. HDMI port of GPU to HDMI port of display

		c. Audio-out of display to line-in of chipset For GPUs that have on-board audio controller: The loopback for this test would be as follows: a. HDMI port of GPU to HDMI port of display b. Audio-out of display to line-in of chipset
85	HdmiCrcTest	HDMI test. See section 9.1.
86	CheckHdmi	HDMI test. See section 9.1.
87	CudaMatsTest	Deprecated and replaced by Test 143.
88	CudaLrfTest	This tests the allocation of thread local variables in CUDA. These are allocated from the local register file (LRF).
89	CudaGrfTest	This tests the allocation of shared variables in CUDA. Variables are shared in the sense that all the threads within a given cooperative thread array can access the value of the variable. Such variables are allocated from the global register file (GRF).
90	FbioLinkTest	This is a simple FB memory interface test that uses the GPU's built-in FBIO training engine to generate the read/write traffic and count errors. Our other FB tests use either CPU traffic over the PCI-E bus (Mats) or the 2d engine (FastMats, WfMats) or 3d engine (RmStress, GLStress). In theory the built-in FBIO training engine should be usable by runtime resman operations to adapt a board on each boot. We're prototyping such adaptive training here. Before the FBIO link training engine starts the test operation, it blocks all other memory clients and swaps in a whole alternate set of "tunable" FBIO registers: read and write strobe timing and voltage-ref. This allows tuning to proceed to fairly extreme values without corrupting instance memory.
91	CudaDgemmTest	Deprecated and replaced by Test 190
92	GLStressPulse	This test runs a reduced-resolution version of GLStress with pauses between bursts of frames, to produce an 11khz 50% duty cycle load on the power supply. It sweeps its pulse frequency rather than using a fixed frequency. The actual frequencies vary with board performance, but GT200 sweeps over about 1khz to

		70khz.
93	NewWfMatsShort	An alternate run of NewWfMats (see test 94) which runs only the blit loop with no CPU loop at all.
94	NewWfMats	NewWfMats divides the FB memory up into many 2d "boxes", each 1024 pixels wide by BoxHeight lines high. These are put into two lists, one for CPU read/write (as in the MATS test) and the other for the gpu blit loop. This is an updated version of the WfMats test for new GPUs. New features include bandwidth reporting (GB/sec in the blit loop) and new controls of CPU box-list such that test time will be about the same for any board regardless of FB size. Each box-list is shuffled randomly so that our read/write operations jump around in the FB address space, allowing us to catch addressing errors that sequential operations would miss. The duration of the test is controlled by the size of the CPU box list. By default, the CPU box list contains about 1/8th of all FB memory. The Coverage property reduces this ratio, reducing test time. The End property limits the FB memory in total, also reducing test time. When each cpu box has been read/written to each pattern in the CpuPattern list (default is 4 patterns: 0x00000000, 0xffffffff, 0xaaaaaaaa, 0x555555555), the blit loop is stopped and the blit boxes are checked for errors. The blit loop boxes are each initially filled with a different pattern, by default using all 29 patterns supported by the PatternClass object. Each time the blit loop runs, each box is copied to the next box 29 times so that at the end of the loop each box returns to its original pattern. Errors in reading or writing accumulate for the duration of the test, and are recorded at the end when each box is read back by the CPU and checked against its original pattern. The errors detected by the blit loop cannot be definitively mapped to any particular memory cell, since they accumulate across many boxes. But WfMats is careful to start each box at an address that is a multiple of the partition-swizzling interleave, so that we know the most important info about the error: partition, burst-order, byte-lane, data-bit.

95	GLStressFbPulse	This is a variant of GLStressPulse, in mods 177.5 and later. It is intended to catch power-supply problems in the FB section of the board. This is achieved by using smaller minimum chunks of drawing to hit a higher pulse frequency and tuning the GLStress options to increase the framebuffer interface workload. Be aware that GLStressFBPulse is still experimental and is not run by default.
97	GLRandomTpc	Runs the GLRandom_hwtest, which issues random graphics operations through the OpenGL driver. This checks CRC's per texture processing cluster (based on screen coordinates) and allows the test to isolate any failures to a particular TPC.
98	CudaStress	CudaStress (test 98) is a CUDA based compute stress test. It is intended to keep the GPU busy with minimal CPU loading (to avoid being CPU bound even when run multi-threaded on host side). It has LoopMs and KeepRunning test-time controls like GLStress, and is good for running in the background behind other tests for context-switching stress. It uses a fairly small FB memory footprint, so should be mostly GPU bound rather than FB bound.
99	DPLoopback	This test is similar to HDMILoopback test except for 2 things: 1. Instead of HDMI port of the GPU, we use the DP port to carry audio. 2. You need a display which enables audio over Display Port(which is currently hard to find).
100	CpyEngTest	This test focuses on testing the CopyEngine by performing a series of copies between surfaces and checking the results in a manner similar to GpuDma. The surfaces can vary in size and must be in either frame buffer or system memory. They can have either block linear or a pitch layout. Once a copy is completed, if either the source or destination surface is in the frame buffer, that surface is copied again to system memory. This is done to speed up the integrity check. The source and destination surfaces (or their copies in system memory) are then compared to make sure the copy was performed correctly. If an error is detected and we have performed the secondary copies

		(moving a surface from frame buffer to system memory to speed up the check), then we will also check the surfaces in the frame buffer to see if the secondary copies are at fault.
101	Elpg	This is the most basic MODS power gating test based on PMU messages . First, MODS allocates graphics, video and VIC engines (if supported) and waits for the PMU to signal power gate. The test then attempts to check whether power gating can be disengaged when MODS attempts to do a privileged register read or push buffer flush on the corresponding engines. For chips that supports power rail gating, this will also attempt to make sure that the chip can enter and exit power rail gating.
102	DispClkStatic	Iterate through all available dispclk perf points and run the display tests (4, 7, 11) at each disp clock point
103	IntAzalia Loopback	Some GPUs starting with GT21x have an onboard azalia audio controller. This test makes sure that that azalia controller is working by creating a loopback between different codecs. It sends an output stream through one codec and will try to receive through another codec. It will then compare the input stream to output stream. If the stream cannot be read or was corrupted, the test fails.
104	PcieLinkTest	This test stresses ASLM (link width change) by changing the PEX link width and throws bursty data on the PEX bus. The test checks for whether the link width change is successful, determines the correctness of the data transfer to sysmem and FB, and also checks whether PCIE errors occurred during the test.
105	I2CSTest	The GPU's internal sensor can act as a slave I2C device. On a system that has another master (happens sometimes in notebook), Gpu temperature can be read through I2C read on the I2CS port. This test checks out if this interface is functional Requirements: a special board is required. The test will attempt to issue I2CS read by issuing read through the I2C port. As a result, the board must have a loopback between the I2CS port and the GPU's I2C port. We try to verify that the value read by I2C -> I2CS is the same

		as the value read back through RM->register reads (Subdev.Thermal.ChipTempViaInt).
106	KFuseSanity	This checks whether valid HDCP keys were blown into the fuse block and verifies that the keys are not all zero. It also confirms that the CRC of the KFuses is correct
107	PStateTest	This is a legacy PState switching test. (A PState is a combination of voltages and clocks and may include some other power-saving features.) The does random PState switches (picked from the available pstates in the perf table) while doing bursty DMA transfers between system memory and frame buffer memory. The test checks the integrity of the data transfers and whether PCIE errors accumulated during the test. In addition, since link width change and link speed change are tied to pstate switches, this test will also attempt to verify that the correct link speed and link width are set for each pstate change. For PState 2.0 and GPU Boost systems, this test has be replaced by test 145.
108	LineTest	This is a verification test to detect reads-passing-writes bugs in corelogic hardware.
110	CudaBoxTest	This is a variant of test 3 (MatsTest) that uses CUDA instead of CPU “dumb framebuffer” accesses to exercise memory.
111	CudaByteTest	This is a variant of test 18 (ByteTest) that uses CUDA instead of CPU “dumb framebuffer” accesses to exercise memory.
112	CudaMarchTest	This is a variant of test 52 (MarchTest) that uses CUDA instead of CPU “dumb framebuffer” accesses to exercise memory.
113	CudaPatternTest	This is a variant of test 70 (PatternTest) that uses CUDA instead of CPU “dumb framebuffer” accesses to exercise memory.
114	CudaAppleMOD3 Test	This is a variant of test 76 (AppleMOD3Test) that uses CUDA instead of CPU “dumb framebuffer” accesses to exercise memory.
115	CudaAppleAddr	This is a variant of test 74 (AppleAddrTest) that uses CUDA

	Test	instead of CPU “dumb framebuffer” accesses to exercise memory.
116	CudaAppleKH Test	This is a variant of test 75 (AppleKHTest) that uses CUDA instead of CPU “dumb framebuffer” accesses to exercise memory.
117	ClockPulseTest	This test programs the GPU's thermal-slowdown feature to “pulse” the effective “gpcclk” speed and thus provide a stress to the power supply. This test only works on the hardware slowdown part of the subsystem we know as the thermal slowdown. It is intended to be run as a background test, while graphics tests run in other threads, to detect any data corruption caused by power-supply glitches. For example you might run it this way: `mods gputest.js -engr -test 2 -bgtest 117`
119	CudaRandom	This test uses CUDA to test out all of the single & double precision mathematically operations for a given compute capability. It is designed to verify consistency not accuracy of these operations.
122	ElpgGraphics Stress	This test toggles graphics ELPG while running GLStress. This is to make sure that engaging power gating has no effect on the correctness of graphics operations. Through command line options, this test can additionally toggle Video ELPG in the background. This creates extra noise and larger change of di/dt on the power rail.
123	NewWfMatsBus Test	This is a memory test that tries to determine if memory failures occur on read or on writes. The amount of IO done by the GPU is deterministic (and very small!). The CPU does no IO at all. It follows this procedure: set dramclk to WriteSpeed fill gpu-loop boxes with initial patterns (via sysmem->FB blits) set dramclk to ReadSpeed read back (via FB->sysmem blits) and verify each gpu-loop

124	ElpgVideoStress	This test engages video ELPG in the background while running MSDEC (test 25). The purpose is similar to test 122. It can additionally toggle Graphics ELPG in the background as well. This creates noise and larger change of di/dt on the power rail.
125	DeepIdleStress	This test runs a version of GLStress that periodically forces a transition into the Deep Idle low power state. The graphics operations generated by the OpenGL driver force the transition out of the low power state. This test requires the VBIOS to support P-State 12, and the upstream bridge device connected to the GPU to support L1 ASPM.
126	GLRandomOcg	Test the internal OpenGL shader compiler by creating a large amount of randomly generated vertex/geometry/fragment programs and then issue random graphics operations through the OpenGL driver. This test is a derivation of test 16. The major difference is that a new set of random shaders are created at the start of each loop instead of each frame. This is a consistency test not an accuracy test.
127	CudaColumnTest	FB memory test for long time retention of data in DRAM cells with sparse write changes. Designed to expose spurious bit flips correlated to refresh cycles and content of DRAM.
128	DeepIdleVETest	This test validates nvDPS and Deep Idle Video Enabled functionality. Random rectangles are rendered to the screen via the 2D engine and periodically the rendering is paused. When this occurs the nvDPS hardware detects a lack of screen activity and signals entry into the Deep Idle Video Enabled state (a low power state with display enabled, and a forced lower refresh rate). This test has the same requirements as DeepIdleStress, but in addition it also requires a LVDS or eDP display.
130	GlrA8R8G8B8	Test the 3-D graphics engine by issuing random graphics operations through OpenGL driver. It uses normal 32-bpp color/Z, i.e. a8r8g8b8 & s8d24
131	GlrR5G6B5	Directed OpenGL test for normal 16-bpp color/Z, i.e. r5g6b5 & d16
132	GlrFsaa2xQx	Directed OpenGL test for 32-bpp color/Z, 2x full-scene-anti-aliasing

133	GlrFsaa4xGs	Directed OpenGL test for 32-bpp color/Z, 4x full-scene-anti-aliasing
135	GlrMrtRgbU	Directed OpenGL test for 3-way Multi-Render-Target 32,24,16 [r8g8b8a8,r8g8b8,r5g6b5] & 32-bpp Z
136	GlrMrtRgbF	Directed OpenGL test for 2-way Multi-Render-Target floating-point 64,128 [rgba_F16, rgba_F32] & 32-bpp Z
137	GlrY8	Directed OpenGL test for 8-bpp color (GL_INTENSITY8) & 32-bpp Z
138	GlrFsaa8x	Directed OpenGL test for 32-bpp color/Z, 8x full-scene-anti-aliasing
139	GlrFsaa4v4	Directed OpenGL test for 32-bpp color, 64-bpp Z, 4v4 VCAA full-scene-anti-aliasing
140	GlrFsaa8v8	Directed OpenGL test for 32-bpp color, 64-bpp Z, 8v8 VCAA full-scene-anti-aliasing
141	GlrFsaa8v24	Directed OpenGL test for 32-bpp color, 64-bpp Z, 8v24 VCAA full-scene-anti-aliasing
142	MultiCellFlipTest	This is a directed test which targets specific single-bit failures found on Hynix memory chips. It attempts to minimize modifying adjacent cells in the same row when testing target cells by limiting modifications to dwords from four adjacent columns (burst length) across all internal banks, external banks, partitions and lanes. It also modifies a few rows simultaneously, because it was discovered that row switching increases failure rate.
143	NewCudaMats	This is an improved CUDA-based MATS test for testing framebuffer memory. Device memory is broken into chunks, each of which is stressed by an individual CUDA thread. The test fills the memory with every 32-bit pattern from the chosen pattern set in both ascending and descending directions.
144	CudaMatsPatCombi	CudaMatsPatCombi is a test based on NewCudaMats which goes through various combinations of pattern pairs. Some pattern pair combinations are more stressful at exciting single

		bit memory errors than others.
145	PerfSwitch	This test attempts to switch between available voltage-frequency points while running another mods test in foreground. There are two modes for this test: a. PerfJumps (default mode): The test jumps through the inflection points on the V-F curve b. PerfSweep: Sweep through the V-F curves The default foreground test is GLStress (test 2). In case of a failure, the error code specifies the foreground test number. This test is a replacement for PStateTest (test 107) on GPUs supporting GPU Boost.
146	PexBandwidth	This test uses CopyEngine to saturate the PCIE bus
147	GpuGc6Test	Test for GC6 feature. Enter GC6 and exit G6 by various wakeup events. In each loop, verify FB is not corrupted
148	GlrA8R8G8B8Sys	Directed OpenGL test for 32-bpp color Z, with render to System Memory instead of Frame Buffer
150	MMERandomTest	This is a test of the Method Macro Expander, which is a unit at the front end that can generate pushbuffer methods programmatically via a small simple language. Random MME programs are generated and their output is routed to a surface (rather than to host as pushbuffer methods). This output is then check against the output of a software MME simulator for consistency
151	Bar1Remapper Test	This test validates that the BAR 1 remapper hardware functions correctly by creating randomized block linear surfaces with known data and then reading them back in a pitch linear fashion via the CPU with the BAR 1 remapper correctly configured.
152	GLStressNoFB	GLStress tuned for much reduced FB bandwidth
153	GLPowerStress	A more stressful version of GLStress (test 2)

154	CudaL2Mats	This tests validates the L2 cache on Fermi and newer GPUs. The test monitors the number of hits and misses to the L2 cache. In order for the test to pass, the L2 misses must be under AllowedMissPercent, which defaults to 10%.
155	EccFbTest	This is a test of Frame Buffer ECC logic on ECC-enabled boards.
156	EccL2Test	This is a test of L2 cache ECC logic on ECC-enabled boards.
157	NewWfMatsMemToMem	This is a variant of test 94 (NewWfMats) that uses the "memory to memory format" engine rather than the "2D rendering engine" to do framebuffer->framebuffer memory copies. This engine is less efficient and makes the test run slower, but it is useful for isolating framebuffer problems on GPUs where the graphics pipeline is not working correctly.
161	NewWfMatsCEOnly	This is a variant of test 94 (NewWfMats) that uses the Copy Engine rather than the "2D rendering engine" or "memory-to-memory format" to do framebuffer->framebuffer memory copies.
170	GpuPllLockTest	Ensure that NVPLL/HPLL/SPPLL can be locked properly
174	CheckPwrSensor	This test validates that the power sensors on the board matches the description in boards.js
175	GpuResetTest	This test validates the suspend resume functionality of the GPU. Option for reset via XVE or cold reset (like Optimus)
178	WfMatsBgStress	This test runs WfMats on Copy Engine while running GLStress on the 3D engine. This hybrid test is yet another way to stress the framebuffer interface.
180	NewWfMats Narrow	Runs WfMats in narrow mode. This causes WfMats blits to be broken into one-pixel wide blits. This causes lower bandwidth, but better exercises the FBIO byte-enable lines.
185	CudaRadixTest	Stress the GPU by using radix sort algorithm by Duane Merrill
186	GsyncLink	Using Gysnc device to test quality of DP link, iterating all SORs to ensure XBAR functionality.

187	CudaMatsShmooTest	This test is a variant of test 87 (CudaMatsTest) that iterates with different input parameters to find the most stressful configuration for a given board.
189	GsyncMem	Memory test for the Gsync module in monitors. It reads pre/post memory for several frames.
190	DPStressTest	This is a CUDA based double precision test. On some Tesla system, this test was found to be more stressful than GLStress
198	CudaStress2	CudaStress2 is a variant of test CudaStress (test 98). It is tuned for more stress on Fermi and newer GPUs.
199	CudaLinpackTest	Runs optimized DGEMM based algorithm that tress the GPU similar to Linpack.
200	CudaLinpackSgemm	Variant of CudaLinpackTest (199). Uses single precision.
202	GLRandomGCx	GLRandom test with GC5/6 bubbles
205	I2MTest	This tests the InlineToMemory class on Kepler GPUs.
206	PanelCheck	Automated NVSR panel certification test
208	TCONStress	NVSR TCON stress test
210	CudaHgemm	Half precision (fp16) version of CudaLinpack test (199)
211	CudaHgemmPulse	CudaHgemm (210) with pulsing workload
225	MSENCTest	Test for Video Encoder Engine. This test runs four streams with different H.264 coding CAVLC and CABAC.
227	CudaColumnShmooTest	This test loops test 127 (CudaColumnTest) with various parameters in an attempt to find specific types of DRAM faults.
231	GlrLayered	GLRandom for Layered randering

246	NvLinkBwStress	NvLink Bandwidth test. It automatically sends data across all NvLink connections, verify results and ensure expected bandwidth.
247	GpuGc5	Test for GC5 feature. Enter/exit GC5 while verifying the wakeup reason is correct and verify that FB is not corrupted.
275	BoostBaseClockTest	GLStress based test to hit Boost target temperature, clocks, voltage, and fan speeds.
277	NVDECTest	Test for the Maxwell+ NvDec engine
278	NVENCTest	Test for the Maxwell+ NvEnc engine
286	FillRectangleTest	Test for GL_FILL_RECTANGLE_NV with multi-sampled cases
287	MsAAPR	Test GL Path rendering on multisampled render buffers
289	GLPRTir	Test for target independent rasterization
292	CudaLinpackPulse	Variant of CudaLinpackTest (199) with pulsing workload
293	I2cDcbSanityTest	Test to ensure all the I2C devices in the DCB table are stuffed
295	MscgMatsTest	Test to exercise MSCG
296	CudaLinpackPulseSP	Single precision CudaLinpackPulse (292)
298	NVDECGCxTest	Enter and exit GC5/6 while running NvDec test (277)
346	NvLinkBwStressTwod	Variant of NvLinkBwStress (246) using TWOD engine instead of CopyEngine
347	GcxTest	New generation of GC6/5 test – intermix the two power saving states.
999		This test number is reserved for external scripts.

5.0 TEST RESULT

As each test is executed, it is logged to the log file when it begins, and when it ends. By default the log file name is mods.log. The log file name can be changed via the '-l' command line argument to MODS.

When a test begins, the following message is printed to the log file. The portion in brackets is only printed if the –time command-line option is used.

Enter FastMatsTest.Run [Thu Jan 11 14:42:47 2001]

When a test ends, the following message is printed to the log file.

Exit 19083: FastMatsTest.Run golden value miscompare [5.293 seconds]

In this case, 19083 is the error code of the test. "FastMatsTest.Run" is the name of the test. "golden value miscompare" is a description of the error code. The time in brackets is how long the test took to execute. The execution time is only displayed if the –time command-line option is used.

5.1 Error Codes

MODS error codes are now 12 digits long. These include

1. One digit to signify whether this test was
    a. run at static voltages and clocks
    b. switching between inflection points on a V-F curve
    c. Sweeping the V-F curve
2. Three digits to specify additional PState information
    a. PState type:

        0 – explicit

        1 – min

        2 – max

        3 – TDP (Thermal Design Point)

        4 – TURBO (aka Boost or TurboBoost)

        5 – intersect – voltage

        6 – intersect – voltage/frequency point

7 – intersect – PState

8 – intersect – VPState

9 – multiple intersect parameters

b. Intersect Rail:

0 – Not at intersect

1 – NVVDD

2 – NVVDDS

c. Master Clock Domain Used for Intersect:

0 – Not at intersect

1 – At intersect but not applicable

2 – gpcclk

3 – dispclk

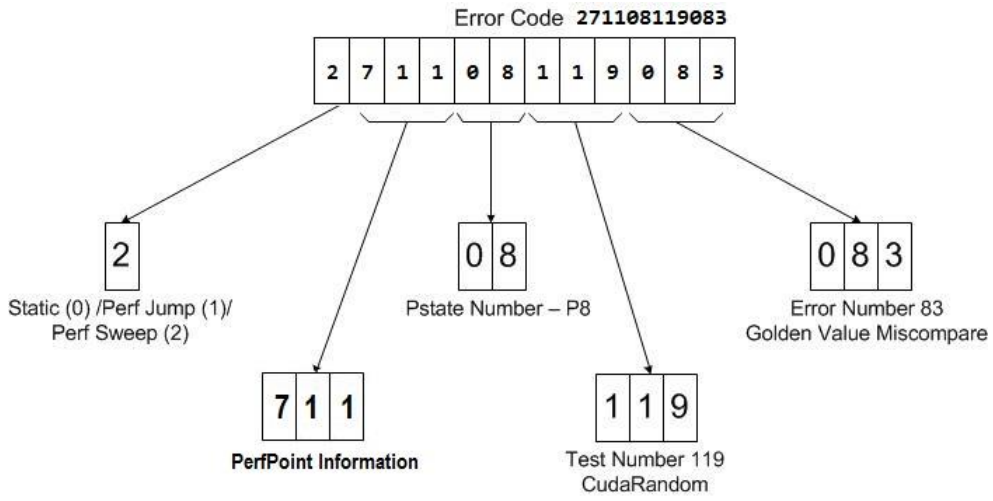
4 – dramclk

5 – pexclk

3. Two digits to represent P-states used during this run
4. Three digit test number
5. Three digit error code

P-states are generally 0, 1, 5 or 8, but others are possible. The test numbers start at 1 and end at 347. Errors are between 1 and 999.

For example, an error code of 271108119083 would mean that the CudaRandom, test 119, failed in PerfSweep function of test 145, while at intersect on NVVDD rail, in p-state 8 with an 83 error, which a golden value miscompare.



The test number 999 is reserved for scripts and tools external to MODS. Error codes like "999123" are not returned by MODS itself.

6.0 DEBUGGING TECHNIQUES

If a card fails, take a look at the log, then attempt to deduce what could be wrong. If the log is very long, attempt to look for keywords like "failure" or "error".

You may attempt to isolate to whether the problem is display related.
Adding -null_display would disable display.

You may attempt to isolate using –test or testspec to find out which test is catching the problem on the graphics card.

You may attempt to isolate whether the problem is perf related. You may wish to try these experiments:

Try lowering dramclk and gpc2clk:

- mods gputest.js –mfg -dramclk 100 -gpc2clk 200
- mods gputest.js –mfg –dramclk_percent 85 –gpc2clk_percent 85

Note: Some DDR Drams require that the dramclk be above a certain frequency for the DLL to work. Furthermore, some products require that you keep dramclk and gpc2clk in less than a 2:1 ratio.

Try looping the test that is failing.

- mods gputest.js -mfg -test 5 -loops 100

Look at the debug-level mods.log output file.

- mods -C gputest.js –mfg
- mods gputest.js –mfg –verbose

- mods –d gputest.js –mfg

If memory tests are failing, you can get extra information on the failure in the log file by using the –matsinfo command-line option.

- mods gputest.js –mfg –matsinfo

7.0 STAND-ALONE MATS

In certain situations MODS cannot initialize the GPU due to marginal frame buffer interface timings or defective memory. In such situations you can try running the stand alone MATS. One will have to boot the GPU as primary in order to run MATS. This utility will do a rudimentary test of the framebuffer. It prints its results to the screen and also to a file named "report.txt." This utility is only available for Linux. Stand-alone MATS produces an output file called "report.txt" that contains data about which framebuffer bits failed.

It is not usually necessary to test the entire framebuffer to collect enough error statistics to be useful. The user can run "mats –c 1" which will test 1% of memory distributed throughout the framebuffer. This is useful because it will complete in a very short time and still produce meaningful debug information in the report.txt file.

8.0 GPU TESTS

This section describes special tests that require unusual configurations or user activity.

8.1 HDMI

MODS includes an HDMI test that uses audio loopback. It requires extra hardware and setup. Since HDMI-audio requires us to send a SPDIF signal into the board, the test requires a motherboard that meets the following requirements:

The motherboard must have an electrical SPDIF-out port.

The GPU being tested must have an embedded Azalia audio controller.

To run the loopback test, you must:

Connect the headphone jack on the HDMI display to the line-in or mic-in jack on the motherboard.

Run either "mods gputest.js -engr -test 84" or "mods gputest.js -engr -test 85". (The latter runs all the usual gputest tests in addition to the hdmi tests.)

There are some known issues due to differences between various motherboards and displays:

1. Try the loopback test without the headphone-jack cable. You should hear a hum as long as the display volume is high enough. This step ensures MODS is correctly driving on audio signal over HDMI.

2. There's no telling which port number is used for the line-in or mic-in jack. Try connecting the audio cable and run MODS with different "-line_in" values (-line_in 0, -line_in 1, etc) until you get a successful loopback -- or at least an error message that says something like "frequency mismatch" rather than "unexpected silence".

3. Finally, you may need to adjust the volume of the display. It needs to be loud enough so that the motherboard can "hear" it, but not so loud that the signal gets clipped. An "unexpected silence" error indicates the volume's too low, while a "frequency mismatch" error indicates the volume's too high. (A good starting point is to run the test while listening to a pair of headphones attached to the display. Find a volume that can be heard but isn't painfully loud.)

8.2 HDCP

Some graphics cards support an encryption protocol called HDCP (High Definition Content Protection). This protocol encrypts data between an HDCP-enabled digital flat panel and an HDCP-enabled graphics card.

The only way to test HDCP is to enable it with an HDCP-enabled display attached. One of the goals of MODS is to enforce a textbook-correct test by default. Therefore, an HDCP test is run automatically on HDCP-enabled cards. The upshot of this is that you must have an HDCP-enabled display attached when testing an HDCP-enabled card or MODS will fail. If the user does not want this behavior, then he or she should explicitly skip HDCP testing using the "-skip 24" command-line argument.

There are three HDCP tests in MODS:

1. A key exchange test. This is done with "mods gputest.js –hdcp_keys". This test does not actually enable HDCP, it only does a key exchange and passes if the exchange was successful. If it passes, the key selection vectors (Aksv and Bksv) are printed in the log file. There are many types of manufacturing faults that cannot be caught by this test.

2. The default HDCP test. This test does the key exchange above, then enables HDCP. If the hardware detects that HDCP was successfully enabled, then the test passes. This test will catch most (but not all) types of manufacturing problems. In particular, there is a rare type of defect that can occur when the key exchange and enabling of HDCP are both successful, but there will be snow on the screen.

3. The interactive test. This is does the key exchange and enables HDCP, then prompts the user to ensure that the display looks OK. The key selection vectors and the HDCP status (pass or fail) are displayed on the screen. This test is enabled with "mods gputest.js –check_displays". See section 9.3 below for more information on interactive display tests.

There are some displays by specific manufacturers that are slow to enable HDCP encryption. If you are having problems with a specific display, try using the following command line arguments individually or in combination:

-hdcp_delay 2000

-hdcp_timeout 5000

8.3 Interactive Display Testing

MODS supports several interactive display tests. There are some types of hardware faults that can only be detected with an interactive test. This is a list of interactive display tests and the command lines to enable them:

mods gputest.js -check_display	Display a slanted red, white and blue pattern on the primary display and prompt the user if it is OK.
mods gputest.js -check_display_bar	Display vertical bars on the primary display and prompt the user if it is OK.
mods gputest.js -check_display_bars	Display vertical bars iteratively on all possible display combinations and prompt the user if each one is OK.
mods gputest.js -check_displays	Display a slanted red, white and blue pattern on all possible display combinations and prompt the user if each one is OK.
mods gputest.js -check_fp_gray	Display various black, white and gray geometry on all detected DFPs one at a time and prompt

	the user if each one is OK. The pattern displayed has been demonstrated to detect some types of TMDS timing and noise issues.
mods gputest.js -check_fp_stripes	Display various a special TMDS stress pattern on one at a time and prompt the user if each one is OK. The pattern displayed has been demonstrated to detect some types of TMDS timing and noise issues.

9.0 PERFPOINT TESTING AND TEST SPECIFICATIONS

MODS defaults to iterating over many "PerfPoint"s. A PerfPoint is a set of performance (clock) settings.

By default, -mfg will run tests at each memory-clock setting (pstate) twice: once at max shader clocks and voltage, and again at min voltage for that pstate. The PerfPoint testing infrastructure is built on top of test specifications. Test specifications are lists of tests that control when a given test is run. A simple example of a test specification is shown below:

```
function addEngrStressTests(spec, perfPoints)
{
    spec.AddTests(["RmStress"
                   ,"WfMatsMedium"
                   ,"GLStress"
                   ,"GLStressPulse"
                   ,"NewWfMatsNarrow"
                   ,"GLRandomCtxSw"
                  ]);
}
```

We can also control the parameters for each test. We have two additional tests that allow the user to set the PerfPoint and to run user defined functions.

```
function addSltTests(spec, perfPoints)
{
    spec.AddTest("RunUserFunc", {"UserFunc": SetFanSpeed, "PctOfMax": 42});
    spec.AddTest("SetPState", {"InfPts": perfPoints[0]});
    addSltPerPStateTests(spec);
```

```
    spec.AddTest("SetPState", {"InfPts":perfPoints[1]});
    spec.AddTest("RunUserFunc", {"UserFunc": SetFanSpeed, "PctOfMax": 100});
    addSltPerPStateTests(spec);
}
```

The standard test ones are visibile in gpulist.js The following contains examples of how test specifications can be used.

```
function addDvsTests(spec, perfPoints)
{
    spec.AddTests(["EvoSli"
                  ]);
    addEngrTests(spec);
    spec.RemoveTests(["FuseRdCheck"
                      ,"GLStressPulse"
                      ,"GLRandomCtxSw"
                     ]);
}

function addSltTests(spec, perfPoints)
{
    spec.AddTest("RunUserFunc", {"UserFunc": SetFanSpeed, "PctOfMax": 42});
    spec.AddTest("SetPState", {"InfPts": perfPoints[0]});
    addSltPerPStateTests(spec);
    spec.AddTest("SetPState", {"InfPts":[1]});
    spec.AddTest("RunUserFunc", {"UserFunc": SetFanSpeed, "PctOfMax": 100});
    addSltPerPStateTests(spec);
}

function addSltPerPStateTests(spec)
{
    spec.AddTests(["FuseRdCheck"
                   ,"MultiBoardDma"
                   ,"SMRom"
                   ,"ElpgGraphicsStress"
                   ,"DeepIdleStress"
                   ,"RmStress"
                   ,"WfMatsMedium"
                   ,"GLStress"
                   ,"GLStressPulse"
                   ,"NewWfMatsNarrow"
                   ,"GLRandomCtxSw"
                  ]);
    addEngrComputeTests(spec);
    spec.AddTests(["CheckFbCalib"
```

```
                    ]);
}

function SetFanSpeed()
{
    var rc;
    var g = this.BoundGpuSubdevice;
    CHECK_RC(g.Thermal.SetCoolerPolicy (Thermal.CoolerPolicyManual));
    CHECK_RC(g.Thermal.SetFanSpeed(this.PctOfMax));
    Out.Printf(Out.PriHigh, "SetFanSpeed: %d pct, trying for %d pct \n",
               g.Thermal.FanSpeed, this.PctOfMax);
    return OK;
}
```

There are two functions to allow users to write out and read back in their own test specifications.

```
mods gputest.js -mfg -savespec {filename} ...
```

-savespec will save the identified specification to a filename specified by the user.

```
mods gputest.js -readspec {filename} ...
```

-readspec uses the test specification as defined in filename. If readspec is is used, then do not specify a test specification (-mfg, -slt, etc); this will override the specification in the user defined file.

Using these two functions, an enduser can write out a test specification, modify it as they see fit, and then use it for their testing.

10.0 CONCURRENT TESTING

Concurrent MODS gives a user the ability to run multiple MODS tests simultaneously on one or multiple GPUs. These tests can either be setup manually on the command-line or MODS can try to run as many tests concurrently as possible.

Within concurrent MODS, there are two different types of threads: foreground and background. A background thread is controlled by a foreground thread and will be stopped once the foreground thread finishes its execution. The specific details of these background threads can be controlled through the various command-line arguments listed below.

Note: By default, concurrent MODS is not turned on. You must specify specific command-line arguments listed below to enable concurrent MODS.

10.1 Command-line Arguments

If an argument is listed as "device sensitive", that means its placement relative to a "-dev Y" is important. If the argument comes before any "-dev Y" options, it will be applied to every GPU. If it comes after a "-dev Y" argument, it will only be applied to that specific device.

Any MODS arguments that are NOT device sensitive, must come before any "-dev Y" are used on the command-line.

The last "-dev Y" used on the command-line will set the primary GPU tested by MODS unless you're running with the "-concurrent_devices" argument.

-bgfunc X

A device sensitive argument that will run the given function X as a background thread. Note, this function needs to call "this.SignalSetupCompleteAndWait()" at some point. It should also check "this.KeepRunning" to know when to stop.

```
function foo()
{
   // Initial setup
   this.SignalSetupCompleteAndWait();
   do
   {
      // Do stuff
   } while (this.KeepRunning);
}
```

-bgtest X

A device sensitive argument that will run test number X as a background thread.

Example, to run Random2D as a background thread on device 1 and run GpuDmaTest as a background thread on all devices:

mods gputest.js -mfg -bgtest 61 ... -dev 1 -bgtest 58 ...

-bgtest_flags X Y

Similar to -bgtest, but takes a comma-separated list of flags for the second argument

Current flags are

disp - Display the background test

roe - Run On Error, continue running the background test even if it fails

mods gputest.js -mfg -bgtest_flags 16 disp,roe ...

-concurrent_devices

This argument will run the tests (the set of tests can be different per GPU) on each GPU in the system, concurrently.

-threadid

This argument simply prepends the ID of the calling thread to the beginning of each line of text in the log. This is useful for seeing which test or GPU printed which line.

You will see the name and ID of each thread listed with +++ thread_name thread_ID +++ in the log

+++ main_tests 10 +++

+++ bg_tests_0 11 +++

10.2 Command-line Examples

Run test 16 as a "background" thread on device 1 with display and ignore errors, run the full MODS suite minus a few tests on device 0. Also print thread information. (Note: since “-dev 0” comes last on the command-line, it will be the primary/foreground GPU tested by MODS.)

- mods gputest.js -mfg -threadid -dev 1 -bgtest_flags 16 disp,roe -dev 0

-skip 24 -skip 17

Run the full MODS suite on all GPUs in the system concurrently

- mods gputest.js -mfg -concurrent_devices

Run Random2d and GpuDma on device 0 sequentially, Run Random2d, GLStress and TurboCipher on device 1 concurrently. Have both GPUs running their set of tests at the same time.

- mods gputest.js -mfg -concurrent_devices -test 58 -dev 0 -test 61 -dev 1

-test 2 -test 79 -concurrent

11.0 ERROR CODES

1 exit
2 software error
3 function is not supported
4 did not install singleton
5 bad command line argument
6 on entry failed
7 bad help string
8 bad parameter passed to function
9 cannot allocate memory
10 cannot open file
11 file does not exist
12 failed while reading a file
13 cannot log method
14 cannot log functions
15 method is still being logged
16 user aborted the script
17 could not create JavaScript engine
18 could not create a JavaScript method
19 could not create a JavaScript object
20 could not initialize the JavaScript standard classes
21 script failed to execute
22 script failed to compile and execute
23 could not compile file
24 cannot convert integer to a jsval
25 cannot convert jsval to an integer
26 cannot convert boolean to a jsval
27 cannot convert jsval to a boolean
28 cannot convert jsval to a float
29 cannot convert float to a jsval
30 cannot convert jsval to a string
31 cannot convert string to a jsval
32 cannot convert jsval to an array
33 cannot convert array to a jsval
34 cannot convert jsval to an object
35 cannot convert jsval to a function
36 invalid object property
37 cannot enumerate object
38 cannot get element
39 cannot set element
40 bad format specification
41 cannot hook interrupt
42 did not initialize resource manager
43 did not initialize resource manager hardware abstraction layer
44 did not map device in to resource manager
45 did not initialize client
46 NVRM invalid base
47 NVRM invalid class
48 NVRM invalid client
49 NVRM invalid device
50 NVRM invalid event
51 NVRM invalid flags
52 NVRM invalid index
53 NVRM invalid limit
54 NVRM invalid object buffer
55 NVRM invalid object error
56 NVRM invalid object new
57 NVRM invalid object old
58 NVRM invalid object parent
59 NVRM invalid offset
60 NVRM invalid param struct
61 NVRM insufficient resources
62 NVRM invalid function
63 NVRM invalid owner
64 NVRM invalid heap
65 NVRM multiple memory types
66 NVRM object has children
67 NVRM object in use
68 NVRM operating system error
69 NVRM protection fault
70 NVRM dual link in use
71 NVRM gpu not full power
72 NVRM invalid dma specifier
73 NVRM no free memory
74 NVRM generic error
75 NVRM irq not firing
76 error occurred while preprocessing file
77 timeout error
78 unsupported depth
79 unsupported surface offset
80 unsupported color format
81 unsupported DOS configuration (EMM is loaded or XMM is missing)
82 stored CRC/Checksum not found
83 CRC/Checksum miscompare
84 file parse error
85 syntax error in FancyPicker configuration
86 incorrect file format
87 failed while writing a file
88 failed to copy memory
89 bad data in trace file or unsupported trace file feature
90 unsupported 3D primitive class
91 failed to render a solid rectangle
92 cannot disable user interface
93 cannot enable user interface
94 memory location must be one of Memory::Fb, Memory::Coherent or Memory::NonCoherent
95 golden value miscompare in Z buffer
96 test configuration has invalid channel type, try TestConfiguration.DmaChannel
97 unexpected device interrupts
98 cannot initialize OpenGL
99 unknown GL error
100 OpenGL error INVALID_ENUM
101 OpenGL error INVALID_VALUE
102 OpenGL error INVALID_OPERATION
103 OpenGL error STACK_OVERFLOW
104 OpenGL error STACK_UNDERFLOW
105 OpenGL error OUT_OF_MEMORY
106 OpenGL error TABLE_TOO_LARGE
107 OpenGL util error INVALID_ENUM
108 OpenGL util error INVALID_VALUE
109 OpenGL util error OUT_OF_MEMORY
110 OpenGL util error INVALID_OPERATION
111 OpenGL util error NURBS_ERROR(n)
112 OpenGL util error TESS_ERROR(n)
113 OpenGL util error TESS_MISSING_BEGIN_POLYGON
114 OpenGL util error TESS_MISSING_BEGIN_CONTOUR
115 OpenGL util error TESS_MISSING_END_POLYGON
116 OpenGL util error TESS_MISSING_END_CONTOUR
117 OpenGL util error TESS_COORD_TOO_LARGE
118 OpenGL util error TESS_NEED_COMBINE_CALLBACK
119 RestartPointLoops must be > 0
120 A description for this hardware could not be found.
121 This GPU has an invalid configuration
122 Invalid encryption key

123 Decompressed data differs from expected results.
124 Invalid InfoROM
125 No display device is connected
126 CRC capture failed
127 Vbios Certificate Error
128 Invalid input
129 Invalid input (driver level)
130 Invalid input (test level)
131 cannot allocate event
132 Robust channel Unexpected Error
133 HDCP did not operate properly
134 EDC detected a memory-bus error
135 Encryption and/or decryption of data failed
136 Request for Power state change failed.
137 invalid window
138 A read/write to a register failed.
139 Acceptable temperature limits exceeded or the thermal sensor is broken or miscalibrated
140 Unused error code 140
141 The only devices found in the system are obsolete
142 Display mode is not possible
143 PCI Express bus error
144 CUDA error
145 cuInit failed
146 cuDeviceGet failed
147 cuCtxCreate failed
148 cuFuncGetByName failed
149 A specific test was requested to run, but was skipped.
150 No tests were run.
151 Primary surface already in use
152 USB invalid RhPort
153 Display HW in use by another test
154 compute test failed
155 Test exceeded the expected threshold
156 Test exceeded the maximum number of allowed memory leaks
157 This board needs to be reflashed with different vbios
158 CRC values are not unique
159 NVRM display is not ready.
160 Resource is reserved by another thread or test
161 NVRM invalid address.
162 USB Reg_Bits not set as expected
163 USB reg not set as expected
164 USB setup packet fail
165 ECC detected a single-bit error
166 ECC detected a double-bit error
167 USB DataIn packet fail
168 USB DataOut packet fail
169 registry key not found
170 registry error
171 incorrect rom version
172 golden check found bad pixel, continuing
173 stored golden values have wrong NumCodeBins
174 golden value miscompare
175 invalid z pitch
176 IRQ not assigned
177 invalid IRQ
178 invalid NV base address
179 invalid NV size
180 invalid FB base address
181 invalid max AGP requests
182 cannot set state
183 invalid AGP request dept
184 invalid AGP data rate
185 cannot set pixel clock
186 cannot set memory clock
187 cannot set graphics clock
188 bad dac
189 invalid channel
190 invalid subchannel
191 bad format
192 put caught up to get
193 invalid ram amount
194 bad memory
195 EDVR: system error
196 ECIC: not CIC or lost CIC during command
197 ENOL: write detected no listeners
198 EADR: board not addressed correctly
199 EARG: bad argument to function call
200 ESAC: function requires board to be SAC
201 EABO: asynchronous operation was aborted
202 ENEB: non-existent board
203 EDMA: DMA hardware error detected
204 EBTO: DMA hardware uP bus timeout
205 EOIP: new I/O with old I/O in progress
206 ECAP: no capability for intended operation
207 EFSO: file system operation error
208 EOWN: Shareable board exclusively owned
209 EBUS: bus error
210 ESTB: serial poll queue overflow
211 ESRQ: SRQ line 'stuck' on
212 ETAB: the return buffer is full
213 ELCK: board or address is locked
214 unknown GPIB Error
215 could not allocate a buffer
216 Could not find the specified device
217 pci bios is not present
218 pci function is not supported
219 pci invalid vendor identification
220 pci device not found
221 pci invalid register number
222 cpuid instruction is not supported
223 cpu does not support MTRR
224 cpu is not supported
225 invalid register number
226 invalid address
227 could not map physical address
228 could not free physical memory map
229 hardware was not initialized
230 invalid graphics aperture base
231 invalid graphics aperture size
232 wrong bios
233 bad NVIDIA chip
234 error occurred while reading or writing serial data
235 could not set environment variable
236 the expected value and the destination memory value do not match
237 unable to set mode
238 specified video mode not found in mode timings table
239 invalid display type
240 invalid tv standard
241 invalid head
242 failed to set image offset
243 failed to disable the cursor
244 feature is not supported in the hardware
245 TIMEOUT: Timeout occurred on WaitSRQ
246 SRQ from Unknown source.
247 Javascript method is not defined
248 Bad SOR - CRC miscompare
249 AUDIO all descriptor entries have buffer
250 AUDIO no valid buffer in descriptor.
251 AUDIO invalid 16bit sample number.

252 CANNOT enable Io or Mem Space.
253 CANNOT enable Bus Master.
254 MemSize detected an invalid framebuffer size.
255 AUDIO not any buffer get freed.
256 MODEM all descriptor entries have buffer
257 Unused error code 257
258 MODEM not any buffer get freed.
259 Golden testname or recname too long.
260 CODEC NOT ready.
261 golden value miscompare in instance memory
262 oven communication error
263 couldn't reach target temperature
264 temperature value not valid
265 CRC error while communicating with oven
266 must first initialize oven
267 PMU device failure, operation attempted failed
268 Invalid Bar(s) assigned to device
269 No Sub Devices found
270 Acoustic test failed, noise too high
271 Sub Device Index Invalid
272 Read parameter differs from expected
273 Clock speed below specified limit
274 Current MODS version doesn't support this Tegra version
275 HW entries have run out
276 HW reports wrong status
277 Error bit set in status register after command was issued
278 Interrupt status differs from expected
279 No free head available
280 Power above specified limit
281 Temperature above specified limit
282 Performance varies from expected value
283 Incorrect OpenGL driver version.
284 unsupported system configuration
285 NVRM buffer too small
286 NVRM reset required
287 NVRM invalid request
288 Power is below specified limit
289 Display underflow detected
290 Unused error code 290
291 Unused error code 291
292 Unused error code 292
293 Data too large.
294 Cannot use loops with PIO channel.
295 Must set a jump point before writing a jump.
296 Subsequent channel writes wrote over jump location.
297 No loop to stop.
298 Usb port not connected to any device
299 Usb Test Fail at configuration
300 AUDIO Test Fail
301 AUDIO Loopback test frequency mismatch
302 Drive test failed
303 MODEM Test Fail
304 MODEM Loopback test frequency mismatch
305 incorrect subsystem id
306 Ism experiment is not complete
307 Timed out waiting for MINI Isms to complete
308 InfoROM not found
309 Unused error code 309
310 Unused error code 310
311 Unused error code 311
312 bad index into FancyPicker array
313 Unused error code 313
314 Unused error code 314
315 Unused error code 315
316 Unused error code 316
317 Unused error code 317
318 Unused error code 318
319 Unused error code 319
320 Unused error code 320
321 Unused error code 321
322 Unused error code 322
323 Unused error code 323
324 Unused error code 324
325 Unused error code 325
326 Generic I2C error
327 TIMER TEST Invalid Counter number
328 TIMER TEST No counter value Returned
329 TIMER TEST timer ticket number doesn't match the expected
330 Audio Invalid Aci Type
331 Hardware does not support this FSAA mode
332 Unused error code 332
333 Unused error code 333
334 Unused error code 334
335 Unused error code 335
336 Unused error code 336
337 Pool CANNOT allocate anymore memory
338 Pool exceed maxim size
339 Pool invalid request size
340 Pool Invalid address to free
341 Buffer mismatch
342 PMU Test Failed
343 Audio Requested channels cannot be enabled
344 Unused error code 344
345 Unused error code 345
346 Unused error code 346
347 The Current Codec doesn't have loopback mode.
348 Unused error code 348
349 Out of date golden file.
350 incorrect chip revision
351 memory not strapped correctly
352 AUDIO Loopback test amplitude mismatch
353 Unused error code 353
354 Unused error code 354
355 Audio Processing Unit timeout
356 Audio Processing Unit CRC miscompare
357 Audio Processing Unit failed to get resources
358 Audio Processing Unit error
359 Each board description must be unique
360 Audio timeout Error
361 Unused error code 361
362 Unused error code 362
363 Unused error code 363
364 Audio CODEC power down register has wrong value
365 CRTC FIFO underflow occurred
366 The order of commands in the MPEG stream was not correct
367 Found a bad command in the MPEG stream
368 MPEG hardware sent the wrong number of notifiers
369 Audio Resource Manager initialization failed
370 bad stereo glasses connector
371 Device Register PIO Access not enabled
372 Device Register Memory Access not enabled
373 Device DMA not enabled
374 Not High Speed Device connected to Usb2 port
375 The user determined that the TV quality was unacceptable
376 Unused error code 376
377 Unused error code 377
378 Unused error code 378
379 Unused error code 379

380 Unused error code 380
381 Unused error code 381
382 Unused error code 382
383 Unused error code 383
384 Unused error code 384
385 Unused error code 385
386 Unused error code 386
387 Unused error code 387
388 Unused error code 388
389 Unused error code 389
390 Unused error code 390
391 Unused error code 391
392 Unused error code 392
393 Unused error code 393
394 Unused error code 394
395 Unused error code 395
396 Unused error code 396
397 Unused error code 397
398 Unused error code 398
399 Unused error code 399
400 Unused error code 400
401 Unused error code 401
402 Unused error code 402
403 Unused error code 403
404 Unused error code 404
405 Unused error code 405
406 Unused error code 406
407 Unused error code 407
408 incorrect TV encoder type
409 Unused error code 409
410 Unused error code 410
411 Unused error code 411
412 Unused error code 412
413 Unused error code 413
414 Remote Controller Test Not ALL Key were tested.
415 Remote Controller Test Key Pressed Mismatch expected.
416 Remote Controller Test Register value Mismatch expected.
417 Network is not initialized.
418 Network cannot create socket.
419 Network socket cannot bind to the specified port.
420 Network socket cannot connect to peer.
421 Network socket is not connected.
422 Network socket is already connected.
423 Network read error.
424 Network write error.
425 Network cannot determine host address.
426 A network error has occurred.
427 Unused error code 427
428 Data vector size mismatch expected.
429 Data vector value miscompare with expected.
430 error occurred trying to write a call pushbuffer instruction
431 not enough pushbuffer memory
432 cdrom audio quality was unacceptable
433 avpod audio quality was unacceptable
434 tuner audio quality was unacceptable
435 Unused error code 435
436 Unused error code 436
437 Unused error code 437
438 Unused error code 438
439 vbe call failed
440 wrong vbe signature
441 wrong vbe version
442 Unused error code 442
443 Unused error code 443
444 Unused error code 444
445 Unused error code 445
446 incorrect mode
447 incorrect vga windows
448 File size would become larger tha the implementation can support.
449 File exists but cannot be accessed with given flags.
450 File write followed a nonblocked write before the latter was complete.
451 File argument isn't valid file descriptor or isn't open for writing.
452 File device or resource is busy.
453 No child process.
454 File deadlock.
455 File open with O_CREAT and O_EXCL set but the file already exists.
456 File bad address.
457 File is too large.
458 File operation was interrupted by a signal.
459 File argument not valid.
460 File I/O error
461 The open operation was interrupted by a signal.
462 The process has too many files open.
463 Too many file links.
464 Filename is too long.
465 The system has too many files open.
466 No such device in file operation.
467 No such file or directory.
468 Exec() format error in file operation.
469 The system has run out of file lock resources.
470 Not enough memory for file operation.
471 Not enough disk space left.
472 File function not implemented.
473 File argument is not a directory.
474 Directory isn't empty.
475 Inappropriate I/O control operation.
476 No such device or address in file operation.
477 File operation not permitted.
478 Write to pipe or FIFO that isn't open for reading by any process
479 File on read-only file system and invalid flags are set.
480 Illegal file seek.
481 Invalid process during file operation.
482 Invalid cross-device link during file operation.
483 Unknown file error.
484 golden value miscompare on 2nd GPU
485 golden value miscompare in Z buffer on 2nd GPU
486 timeout waiting for notifier from GPU
487 timeout waiting for notifier from 2nd GPU
488 Cannot access device registers.
489 the memory or frame buffer interface is marginal
490 Cannot set AGP data rate.
491 Cannot set AGP sideband addressing mode.
492 Cannot set AGP fastwrite mode.
493 Couldn't lock on to the input signal.
494 Couldn't lock on to the chroma data.
495 Actual crystal value does not match the strapped crystal value.
496 invalid display mask
497 failed to get image offset
498 Invalid device Id
499 SBIOS test failed
500 A problem has been detected in the array of tests
501 Test failed due to an already-known problem.
502 Invalid Mfgtest test number
503 Invalid Mfgtest test mode

504 Unused error code 504
505 AUDIO Loopback Left and Right Channel Crossed
506 Unused error code 506
507 Unused error code 507
508 Invalid Chip Version
509 Not an NV Device
510 Test Cannot run on this Tegra Chip Version
511 Required chip library interface not found
512 Unused error code 512
513 Unused error code 513
514 Unused error code 514
515 Unused error code 515
516 Unused error code 516
517 Unused error code 517
518 Usb Port mapping value is wrong.
519 Unused error code 519
520 Unused error code 520
521 Number of Channel and number of input mismatch.
522 Unused error code 522
523 Unused error code 523
524 Unused error code 524
525 Unused error code 525
526 Unused error code 526
527 System Control Invalid IO Base.
528 Unused error code 528
529 Unused error code 529
530 Unused error code 530
531 Usb invalid device.
532 Unused error code 532
533 Unused error code 533
534 Unused error code 534
535 Unused error code 535
536 Unused error code 536
537 Unused error code 537
538 Unused error code 538
539 Unused error code 539
540 Unused error code 540
541 Unused error code 541
542 Unused error code 542
543 Unused error code 543
544 Unused error code 544
545 Unused error code 545
546 Unused error code 546
547 Invalid CPU Frequency measured.
548 Unused error code 548
549 Unused error code 549
550 Unused error code 550
551 Unused error code 551
552 Real time clock test failed to restore.
553 Unused error code 553
554 Graphics fifo method error.
555 GPU channel fifo software method error.
556 GPU channel fifo unknown method error.
557 GPU channel fifo channel busy error.
558 GPU channel fifo runout overflow error.
559 GPU channel fifo parse error.
560 GPU channel fifo PTE error.
561 GPU channel fifo idle timeout error.
562 GPU channel instance lookup failure.
563 GPU channel debug single-step.
564 GPU channel missing hardware error.
565 GPU channel software method.
566 GPU channel software notify.
567 GPU channel fake error.
568 GPU channel scan line timeout error.
569 GPU channel vblank callback error.
570 GPU channel software method parameter error.
571 Unused error code 571
572 The required function is not supported by present CODEC.
573 Audio CODEC failure.
574 Unused error code 574
575 Unused error code 575
576 Audio Test Invalid loopback Mode.
577 Could not acquire I2C port.
578 I2C SCL pull-up resistor missing.
579 I2C SDA pull-up resistor missing.
580 The auxiliary power connector is not plugged in.
581 can not generate golden values using an official release
582 gpu stress test found pixel miscompares
583 thermal sensor reports overheating
584 Unused error code 584
585 failed to capture internal TV encoder crc
586 the internal TV encoder is bad
587 Smbus Cannot set DDC base.
588 invalid EDID
589 FramLock Test Check Reg Fail
590 FramLock Test Invalid DispalySync Unit of Invalid Displays
591 Unused error code 591
592 FramLock Test Set display(s) to Master fail
593 FramLock Test Set display(s) to Slave fail
594 FramLock Test Loopback Test fail
595 FramLock Test Sync Test fail
596 FramLock Test Sync Test, User and Auto result mismatch
597 NVRM not supported
598 Unused error code 598
599 fan does not seem to cool the chip
600 Usb failure related to port mapping, port number.
601 Acpi timer failure.
602 NVRM bad channel
603 NVRM timeout
604 the counter overflowed
605 the frequency is incorrect
606 API call never returned
607 Bad compression-tag-ram in GPU
608 Interrupt request line stuck asserted
609 Interrupt request mechanism does not work
610 Unused error code 610
611 Unused error code 611
612 Invalid value for Tegra configuration variable(s).
613 Invalid Tegra configuration filename.
614 Extra golden code miscompare
615 Extra golden code miscompare on 2nd GPU
616 Unused error code 616
617 Unused error code 617
618 Unused error code 618
619 Unused error code 619
620 DLL could not be loaded.
621 Unused error code 621
622 Unused error code 622
623 Unused error code 623
624 Error in VBIOS DCB tables.
625 Unused error code 625
626 Unused error code 626
627 Supplied mode not supported by the display.
628 The framebuffer base address register is too small
629 Memory leak detected.
630 Perfmon was already running an experiment
631 Memory access spans page boundary.
632 Memory access to unmapped page.
633 Write access to read-only page.

634 Read access to write-only page.
635 Unused error code 635
636 could not create a JavaScript property
637 Invalid clock domain specified
638 Perfmon could not be reserved
639 Perfmon was not reserved
640 Unused error code 640
641 MsiTest of BR02 Failed.
642 Atapi Test Error
643 Unused error code 643
644 Unused error code 644
645 Unused error code 645
646 Unused error code 646
647 Unused error code 647
648 Unused error code 648
649 Unused error code 649
650 Bad RAM in the GPU.
651 GPU did not get the expected number of lanes
652 Unused error code 652
653 Unused error code 653
654 Unused error code 654
655 nvrm invalid parameter
656 nvrm too many primaries
657 Unused error code 657
658 memory size mismatch expected
659 wrong number of TPCs detected
660 wrong number of framebuffer units detected
661 memory fragment size mismatch expected
662 wrong number of ROPs detected
663 wrong number of shader pipes detected
664 wrong number of vertex engines detected
665 wrong number of PCI express lanes detected
666 incorrect feature set for this SKU
667 could not set NV_PBUS_FS to the desired values
668 could not meet floorsweeping requirements
669 Requested function not supported by Codec
670 Requested function not supported by Aci
671 Error testing L2 cache
672 Unused error code 672
673 Unused error code 673
674 NVRM object not found
675 NVRM gpu is still busy or possibly hung
676 NVRM card not present
677 NVRM in use
678 NVRM invalid access type
679 NVRM invalid argument
680 Unused error code 680
681 NVRM invalid command
682 NVRM invalid data
683 Unused error code 683
684 NVRM invalid method
685 NVRM invalid pointer
686 Unused error code 686
687 NVRM invalid registry key
688 NVRM invalid state
689 NVRM invalid string length
690 NVRM FB Training Failed
691 method count too large
692 pushbuffer too small
693 Unused error code 693
694 Unused error code 694
695 Unused error code 695
696 Unused error code 696
697 Unused error code 697
698 Unused error code 698
699 Unused error code 699
700 Unused error code 700
701 Unused error code 701
702 Unused error code 702
703 Unused error code 703
704 Unhook ISR failed
705 Unused error code 705
706 Unused error code 706
707 Unused error code 707
708 Unused error code 708
709 selected device is not supported
710 Unused error code 710
711 Msi is not supported for this device
712 Cannot enable Intx in Pci Cfg Space
713 Cannot enable Msi in Pci Cfg Space
714 Cannot disable Intx in Pci Cfg Space
715 Cannot disable Msi in Pci Cfg Space
716 Given Cap. is not supported for this device
717 Sata Loopback Test fail
718 invalid starting number of VPEs and/or SHDs
719 Read parameter differs from expected
720 Measured Jitter exceeded maximum amount
721 Failed genlock
722 Non-GL device on GL board
723 Codec error detected
724 Stream Error Detected
725 Ring Buffer Error Detected
726 Azalia Test failed
727 Unused error code 727
728 Ahci Port Error
729 External drive (hardrive, cdRom, est) error
730 ATA Descriptor table is not initialized
731 Unused error code 731
732 Unused error code 732
733 External device does not support the function
734 External device is not found
735 Unused error code 735
736 Unused error code 736
737 Unused error code 737
738 Unused error code 738
739 Unused error code 739
740 Unused error code 740
741 Unused error code 741
742 Unused error code 742
743 Unused error code 743
744 Unused error code 744
745 Unused error code 745
746 Unused error code 746
747 Unused error code 747
748 Unused error code 748
749 Unused error code 749
750 Unused error code 750
751 Unused error code 751
752 Unused error code 752
753 Unused error code 753
754 Unused error code 754
755 Unused error code 755
756 Unused error code 756
757 Unused error code 757
758 Unused error code 758
759 Unused error code 759
760 GPU channel bus master timeout error.
761 GPU channel display missed notifier.
762 GPU channel MPEG software method error.
763 GPU channel ME software method error.
764 GPU channel VP software method error.
765 Unused error code 765

766 Unused error code 766
767 Unused error code 767
768 Unused error code 768
769 Invalid command for Tegra device/controller
770 Invalid memory(buffer or dtables) for hw
771 Invalid JsObject for test.
772 Invalid setup for Tegra test.
773 Data/buffer mismatch with expected
774 Tegra test fail
775 SLI generic error
776 Invalid argument.
777 Error configuring overclocking
778 Error while testing board
779 Voltage value out of range
780 Peripheral device not found
781 Gpu is already linked in a SLI device
782 Video bridge not present
783 Mismatched GPUs not valid for SLI device
784 GPU channel RC Logging Enabled.
785 GPU channel Semaphore Timeout.
786 GPU channel Illegal Notify.
787 GPU channel fifo FBISTATE Timeout Error.
788 GPU channel VP: Unknown Error.
789 GPU channel Bad Address Accessed Error.
790 GPU channel VP2: Unknown Error..
791 GPU channel BSP: Unknown Error..
792 SEC Error
793 MSVLD Error
794 MSPDEC Error
795 MSPPP Error
796 Fifo: MMU Error
797 PBDMA Error
798 FECS Err: Unimpl Firmware Method
799 FECS Err: Watchdog Timeout
800 GPU channel CE0: Unknown Error.
801 GPU channel CE1: Unknown Error.
802 GPU channel VIC: Unknown Error.
803 GPU channel: Reset Channel Verif Error.
804 GPU channel GR: Fault During Context Switch.
805 OS: Preemptive Channel Removal.
806 OS Indicates GPU Has Timed Out.
807 GPU channel CE2: Unknown Error.
808 GPU channel MSENC: Unknown Error.
809 GPU channel NVENC0: Unknown Error.
810 GPU channel NVENC1: Unknown Error.
811 Unused error code 810
812 Unused error code 811
813 Unused error code 812
814 Unused error code 813
815 Unused error code 814
816 Bad MAC Address Programmed
817 PLL could not be locked
818 Critical Memory failure(used -fail_critical_fb_range)
819 Mods detected an assertion failure
820 ELG call failure or unexpected behavior
821 KD call failure or unexpected behavior
822 GPU Double-bit Error.
823 Reset in progress.
824 Silent running constant level set by registry
825 Silent running level transition due to RC error
826 Silent running stress test failure
827 Silent running level transition due to temperature rise
828 Silent running clocks reduced due to temperature rise
829 Silent running clocks reduced due to power limits
830 Silent running temperature read error
831 Display channel exception
832 FB link training failure
833 FB memory error
834 PMU error
835 SEC2 error
836 PMU Breakpoint
837 PMU Halt error
838 Inforom Dynamic Page Retirement Event
839 Inforom Dynamic Page Retirement Failure
840 NVDEC error
841 FECS Err: Register Access Violation
842 FECS Err: Verif Method Violation

Error codes 900=999 reserved for errors from auxiliary scripts that are not part of the normal MODS build. MODS may reclaim these error codes in the future.

Notice

HDMI

Macrovision Compliance Statement

Trademarks

Copyright