Main page
Table of contents
Getting involved
Wiki news
Random page

Interaction
Help
Contributing
Recent changes
Recent talks
New pages
Statistics
Requests

Tools
What links here
Related changes
Special pages
Printable version
Permanent link
Page information

In other languages
Deutsch
Español
فارسی
Français
Italiano
日本語
Русский
中文（简体）

NVIDIA

This article covers the proprietary NVIDIA graphics card driver. For the open-source driver, see Nouveau. If you have a laptop with hybrid Intel/NVIDIA graphics, see NVIDIA Optimus instead.

Related articles

- NVIDIA/Tips and tricks
- NVIDIA/Troubleshooting
- Nouveau
- NVIDIA Optimus
- PRIME
- Bumblebee
- nvidia-xrun
- Xorg
- Vulkan

Contents [hide]

1 Installation
  1.1 Unsupported drivers
  1.2 Custom kernel
  1.3 DRM kernel mode setting
    1.3.1 Early loading
      1.3.1.1 Pacman hook
  1.4 Hardware accelerated video decoding
  1.5 Hardware accelerated video encoding with NVENC
2 Xorg configuration
  2.1 Automatic configuration
  2.2 nvidia-settings
  2.3 Manual configuration
    2.3.1 Minimal configuration
    2.3.2 Disabling the logo on startup
    2.3.3 Overriding monitor detection
    2.3.4 Enabling brightness control
    2.3.5 Enabling SLI
  2.4 Multiple monitors
    2.4.1 Using nvidia-settings
    2.4.2 ConnectedMonitor
    2.4.3 TwinView
      2.4.3.1 Vertical sync using TwinView
      2.4.3.2 Gaming using TwinView
    2.4.4 Mosaic mode
      2.4.4.1 Base Mosaic
      2.4.4.2 SLI Mosaic
3 Wayland
4 Tips and tricks
5 Troubleshooting
6 See also

Installation

Warning: Avoid installing the NVIDIA driver through the package provided from the NVIDIA website. Installation through pacman allows upgrading the driver together with the rest of the system.

These instructions are for those using the stock `linux` or `linux-lts` packages. For custom kernel setup, skip to the **next** subsection.

1. If you do not know what graphics card you have, find out by issuing:

```
$ lspci -k | grep -A 2 -E "(VGA|3D)"
```

2. Determine the necessary driver version for your card by:

- Finding the code name (e.g. NV50, NVC0, etc.) on Nouveau wiki's code names page or [1].
- Looking up the name in NVIDIA's legacy card list: if your card is not there you can use the latest driver.
- Visiting NVIDIA's driver download site.

3. Install the appropriate driver for your card:

- For GeForce 630-900, 10-20, and Quadro/Tesla/Tegra K-series cards and newer [NVE0, NV110 and newer family cards from around 2010 and later], **install** the `nvidia` package (for use with the `linux` kernel) or `nvidia-lts` (for use with the `linux-lts` kernel) package.
  - If these packages do not work, `nvidia-beta`[AUR] may have a newer driver version that offers support.
- For GeForce 400/500/600 series cards [NVCx and NVDx] from around 2010-2011, **install** the `nvidia-390xx-dkms`[AUR] package.
- For even older cards (released in 2010 or earlier), have a look at **#Unsupported drivers**.

4. For 32-bit application support, also install the corresponding *lib32* nvidia package from the **multilib** repository (e.g. `lib32-nvidia-utils`).

5. Reboot. The `nvidia` package contains a file which blacklists the `nouveau` module, so rebooting is necessary.

Once the driver has been installed, continue to **#Xorg configuration**.

Unsupported drivers

If you have a GeForce 300 series card or older (released in 2010 or earlier), Nvidia no longer supports drivers for your card. This means that these drivers do not support the current Xorg version. It thus might be easier if you use the Nouveau driver, which supports the old cards with the current Xorg.

However, Nvidia's legacy drivers are still available and might provide better 3D performance/stability.

- For GeForce 8/9, ION and 100-300 series cards [NV5x, NV8x, NV9x and NVAx], **install** the `nvidia-340xx-dkms`[AUR] package.
- GeForce 7 series cards and older [NV6x, NV4x and lower] do not have a driver packaged for Arch Linux.

Custom kernel

If you are using a custom kernel, compilation of the Nvidia kernel modules can be automated with DKMS.

Install the `nvidia-dkms` package (or a specific branch). The Nvidia module will be rebuilt after every Nvidia or kernel update thanks to the DKMS pacman hook.

DRM kernel mode setting

To enable DRM (Direct Rendering Manager) kernel mode setting, add the `nvidia-drm.modeset=1` kernel parameter.

Note:

- The NVIDIA driver does **not** provide an `fbdev` driver for the high-resolution console for the kernel compiled-in `vesafb` module. However, the kernel compiled-in `efifb` module supports a high-resolution console on EFI systems. This method requires GRUB or rEFInd and is described in NVIDIA/Tips and tricks#Fixing terminal resolution.[2][3][4].
- Nvidia drivers prior to version 470 (e.g. `nvidia-390xx-dkms`[AUR]) do not support hardware accelerated XWayland, causing non-Wayland-native applications to suffer from poor performance in Wayland sessions.

Early loading

For basic functionality, just adding the kernel parameter should suffice. If you want to ensure it's loaded at the earliest possible occasion, or are noticing startup issues (such as the `nvidia` kernel module being loaded after the display manager) you can add `nvidia`, `nvidia_modeset`, `nvidia_uvm` and `nvidia_drm` to the initramfs according to Mkinitcpio#MODULES.

If added to the initramfs, do not forget to run **mkinitcpio** every time there is a `nvidia` driver update. See **#Pacman hook** to automate these steps.

Pacman hook

To avoid the possibility of forgetting to update initramfs after an NVIDIA driver upgrade, you may want to use a pacman hook:

/etc/pacman.d/hooks/nvidia.hook

```
[Trigger]
Operation=Install
Operation=Upgrade
Operation=Remove
Type=Package
Target=nvidia
Target=linux
# Change the linux part above and in the Exec line if a different kernel is used

[Action]
Description=Update Nvidia module in initcpio
```

```
Depends=mkinitcpio
When=PostTransaction
NeedsTargets
Exec=/bin/sh -c 'while read -r trg; do case $trg in linux) exit 0; esac; done; /usr/bin/mkinitcpio -P'
```

Make sure the `Target` package set in this hook is the one you have installed in steps above (e.g. `nvidia`, `nvidia-dkms`, `nvidia-lts` or `nvidia-ck-something`).

Note: The complication in the `Exec` line above is in order to avoid running *mkinitcpio* multiple times if both `nvidia` and `linux` get updated. In case this does not bother you, the `Target=linux` and `NeedsTargets` lines may be dropped, and the `Exec` line may be reduced to simply `Exec=/usr/bin/mkinitcpio -P`.

Hardware accelerated video decoding

Accelerated video decoding with VDPAU is supported on GeForce 8 series cards and newer. Accelerated video decoding with NVDEC is supported on Fermi (~400 series) cards and newer. See **Hardware video acceleration** for details.

Hardware accelerated video encoding with NVENC

NVENC requires the `nvidia_uvm` module and the creation of related device nodes under `/dev`. Manually loading the `nvidia_uvm` module will not create the device nodes, but invoking the `nvidia-modprobe` utility will. Create `/etc/udev/rules.d/70-nvidia.rules`:

/etc/udev/rules.d/70-nvidia.rules

```
ACTION=="add", DEVPATH=="/bus/pci/drivers/nvidia", RUN+="/usr/bin/nvidia-modprobe -c0 -u"
```

Xorg configuration

The proprietary NVIDIA graphics card driver does not need any Xorg server configuration file. You can **start X** to see if the Xorg server will function correctly without a configuration file. However, it may be required to create a configuration file (prefer `/etc/X11/xorg.conf.d/20-nvidia.conf` over `/etc/X11/xorg.conf`) in order to adjust various settings. This configuration can be generated by the NVIDIA Xorg configuration tool, or it can be created manually. If created manually, it can be a minimal configuration (in the sense that it will only pass the basic options to the **Xorg** server), or it can include a number of settings that can bypass Xorg's auto-discovered or pre-configured options.

Tip: For more configuration options, see **NVIDIA/Troubleshooting**.

Automatic configuration

The NVIDIA package includes an automatic configuration tool to create an Xorg server configuration file (`xorg.conf`) and can be run by:

```
# nvidia-xconfig
```

This command will auto-detect and create (or edit, if already present) the `/etc/X11/xorg.conf` configuration according to present hardware.

If there are instances of DRI, ensure they are commented out:

```
#    Load        "dri"
```

Double check your `/etc/X11/xorg.conf` to make sure your default depth, horizontal sync, vertical refresh, and resolutions are acceptable.

nvidia-settings

The **`nvidia-settings`** tool lets you configure many options using either CLI or GUI. Running `nvidia-settings` without any options launches the GUI, for CLI options see **`nvidia-settings(1)`**.

You can run the CLI/GUI as a non-root user and save the settings to `~/.nvidia-settings-rc` or save it as **xorg.conf** by using the option *Save to X configuration File* for a multi-user environment.

To load the `~/.nvidia-settings-rc` for the current user:

```
$ nvidia-settings --load-config-only
```

See **Autostarting** to start this command on every boot.

Note: **Xorg** may not start or crash on startup after saving `nvidia-settings` changes. Adjusting or deleting the generated `~/.nvidia-settings-rc` and/or **Xorg** file(s) should recover normal startup.

Manual configuration

Several tweaks (which cannot be enabled **automatically** or with **nvidia-settings**) can be performed by editing your configuration file. The Xorg server will need to be restarted before any changes are applied.

See **NVIDIA Accelerated Linux Graphics Driver README and Installation Guide** for additional details and options.

Minimal configuration

A basic configuration block in `20-nvidia.conf` (or deprecated in `xorg.conf`) would look like this:

/etc/X11/xorg.conf.d/20-nvidia.conf

```
Section "Device"
        Identifier "Nvidia Card"
        Driver "nvidia"
        VendorName "NVIDIA Corporation"
        BoardName "GeForce GTX 1050 Ti"
EndSection
```

Disabling the logo on startup

Add the `"NoLogo"` option under section `Device`:

```
Option "NoLogo" "1"
```

Overriding monitor detection

The `"ConnectedMonitor"` option under section `Device` allows to override monitor detection when X server starts, which may save a significant amount of time at start up. The available options are: `"CRT"` for analog connections, `"DFP"` for digital monitors and `"TV"` for televisions.

The following statement forces the NVIDIA driver to bypass startup checks and recognize the monitor as DFP:

```
Option "ConnectedMonitor" "DFP"
```

Note: Use "CRT" for all analog 15 pin VGA connections, even if the display is a flat panel. "DFP" is intended for DVI, HDMI, or DisplayPort digital connections only.

Enabling brightness control

This article or section is out of date.

Reason: Potentially obsolete[5], upstream package also seems to be ancient. (Discuss in **Talk:NVIDIA**)

Add to kernel paremeters:

```
nvidia.NVreg_RegistryDwords=EnableBrightnessControl=1
```

Alternatively, add the following under section `Device`:

```
Option "RegistryDwords" "EnableBrightnessControl=1"
```

If brightness control still does not work with this option, try installing **`nvidia-bl-dkms`**[AUR].

Note: Installing **`nvidia-bl-dkms`**[AUR] will provide a `/sys/class/backlight/nvidia_backlight/` interface to backlight brightness control, but your system may continue to issue backlight control changes on `/sys/class/backlight/acpi_video0/`. One solution in this case is to watch for changes on, e.g. `acpi_video0/brightness` with *inotifywait* and to translate and write to `nvidia_backlight/brightness` accordingly. See **Backlight#sysfs modified but no brightness change**.

Enabling SLI

Warning: Since the GTX 10xx Series (1080, 1070, 1060, etc) only 2-way SLI is supported. 3-way and 4-way SLI may work for CUDA/OpenCL applications, but will most likely break all OpenGL applications.

Taken from the NVIDIA driver's **README** Appendix B: *This option controls the configuration of SLI rendering in supported configurations.* A "supported configuration" is a computer equipped with an SLI-Certified Motherboard and 2 or 3 SLI-Certified GeForce GPUs.

Find the first GPU's PCI Bus ID using `lspci`:

```
# lspci | grep "VGA|3D controller"
```

```
00:02.0 VGA compatible controller: Intel Corporation Xeon E3-1200 v2/3rd Gen Core processor Graphics Controller (rev 09)
03:00.0 VGA compatible controller: NVIDIA Corporation GK107 [GeForce GTX 650] (rev a1)
04:00.0 VGA compatible controller: NVIDIA Corporation GK107 [GeForce GTX 650] (rev a1)
08:00.0 3D controller: NVIDIA Corporation GM108GLM [Quadro K620M / Quadro M500M] (rev a2)
```

Add the BusID (3 in the previous example) under section `Device`:

```
BusID "PCI:3:0:0"
```

Note: The format is important. The BusID value must be specified as `"PCI:<BusID>:0:0"`

Add the desired SLI rendering mode value under section `Screen`:

```
Option "SLI" "AA"
```

The following table presents the available rendering modes.

Value	Behavior
0, no, off, false, Single	Use only a single GPU when rendering.
1, yes, on, true, Auto	Enable SLI and allow the driver to automatically select the appropriate rendering mode.
AFR	Enable SLI and use the alternate frame rendering mode.
SFR	Enable SLI and use the split frame rendering mode.
AA	Enable SLI and use SLI antialiasing. Use this in conjunction with full scene antialiasing to improve visual quality.

Alternatively, you can use the `nvidia-xconfig` utility to insert these changes into `xorg.conf` with a single command:

```
# nvidia-xconfig --busid=PCI:3:0:0 --sli=AA
```

To verify that SLI mode is enabled from a shell:

```
$ nvidia-settings -q all | grep SLIMode
```

```
  Attribute 'SLIMode' (arch:0.0): AA
    'SLIMode' is a string attribute.
    'SLIMode' is a read-only attribute.
    'SLIMode' can use the following target types: X Screen.
```

Warning: After enabling SLI, your system may become frozen/non-responsive upon starting xorg. It is advisable that you disable your display manager before restarting.

If this configuration does not work, you may need to use the PCI Bus ID provided by `nvidia-settings`,

```
$ nvidia-settings -q all | grep -i pcibus
```

```
Attribute 'PCIBus' (host:0[gpu:0]): 101.
  'PCIBus' is an integer attribute.
  'PCIBus' is a read-only attribute.
  'PCIBus' can use the following target types: GPU, SDI Input Device.
Attribute 'PCIBus' (host:0[gpu:1]): 23.
  'PCIBus' is an integer attribute.
  'PCIBus' is a read-only attribute.
  'PCIBus' can use the following target types: GPU, SDI Input Device.
```

and comment out the PrimaryGPU option in your xorg.d configuration,

```
/usr/share/X11/xorg.conf.d/10-nvidia-drm-outputclass.conf
```

```
...

Section "OutputClass"
...
    # Option "PrimaryGPU" "yes"
...
```

Using this configuration may also solve any graphical boot issues.

Multiple monitors

See **Multihead** for more general information.

Using nvidia-settings

The **nvidia-settings** tool can configure multiple monitors.

For CLI configuration, first get the `CurrentMetaMode` by running:

```
$ nvidia-settings -q CurrentMetaMode
```

```
Attribute 'CurrentMetaMode' (hostnmae:0.0): id=50, switchable=no, source=nv-control :: DPY-1: 2880x1620 @2880x1620 +0+0 {ViewPortIn=2880x1620, ViewPortOut=2880x1620+0+0}
```

Save everything after the `::` to the end of the attribute (in this case: `DPY-1: 2880x1620 @2880x1620 +0+0 {ViewPortIn=2880x1620, ViewPortOut=2880x1620+0+0}`) and use to reconfigure your displays with `nvidia-settings --assign "CurrentMetaMode=`*`your_meta_mode`*`"`.

Tip: You can create shell aliases for the different monitor and resolution configurations you use.

ConnectedMonitor

If the driver does not properly detect a second monitor, you can force it to do so with ConnectedMonitor.

```
/etc/X11/xorg.conf
```

```
Section "Monitor"
    Identifier     "Monitor1"
    VendorName     "Panasonic"
    ModelName      "Panasonic MICRON 2100Ex"
    HorizSync       30.0 - 121.0 # this monitor has incorrect EDID, hence Option "UseEDIDFreqs" "false"
    VertRefresh     50.0 - 160.0
    Option         "DPMS"
EndSection

Section "Monitor"
    Identifier     "Monitor2"
    VendorName     "Gateway"
    ModelName      "GatewayVX1120"
    HorizSync       30.0 - 121.0
    VertRefresh     50.0 - 160.0
    Option         "DPMS"
EndSection

Section "Device"
    Identifier     "Device1"
    Driver         "nvidia"
```

```
    Option         "NoLogo"
    Option         "UseEDIDFreqs" "false"
    Option         "ConnectedMonitor" "CRT,CRT"
    VendorName     "NVIDIA Corporation"
    BoardName      "GeForce 6200 LE"
    BusID          "PCI:3:0:0"
    Screen          0
EndSection

Section "Device"
    Identifier     "Device2"
    Driver         "nvidia"
    Option         "NoLogo"
    Option         "UseEDIDFreqs" "false"
    Option         "ConnectedMonitor" "CRT,CRT"
    VendorName     "NVIDIA Corporation"
    BoardName      "GeForce 6200 LE"
    BusID          "PCI:3:0:0"
    Screen          1
EndSection
```

The duplicated device with `Screen` is how you get X to use two monitors on one card without `TwinView`. Note that `nvidia-settings` will strip out any `ConnectedMonitor` options you have added.

TwinView

You want only one big screen instead of two. Set the `TwinView` argument to `1`. This option should be used if you desire compositing. TwinView only works on a per card basis, when all participating monitors are connected to the same card.

```
Option "TwinView" "1"
```

Example configuration:

`/etc/X11/xorg.conf.d/10-monitor.conf`

```
Section "ServerLayout"
    Identifier     "TwinLayout"
    Screen         0 "metaScreen" 0 0
EndSection

Section "Monitor"
    Identifier     "Monitor0"
    Option         "Enable" "true"
EndSection

Section "Monitor"
    Identifier     "Monitor1"
    Option         "Enable" "true"
EndSection

Section "Device"
    Identifier     "Card0"
    Driver         "nvidia"
    VendorName     "NVIDIA Corporation"

    #refer to the link below for more information on each of the following options.
    Option         "HorizSync"          "DFP-0: 28-33; DFP-1: 28-33"
    Option         "VertRefresh"        "DFP-0: 43-73; DFP-1: 43-73"
    Option         "MetaModes"          "1920x1080, 1920x1080"
    Option         "ConnectedMonitor"   "DFP-0, DFP-1"
    Option         "MetaModeOrientation" "DFP-1 LeftOf DFP-0"
EndSection

Section "Screen"
    Identifier     "metaScreen"
    Device         "Card0"
    Monitor        "Monitor0"
    DefaultDepth    24
    Option         "TwinView" "True"
    SubSection "Display"
        Modes          "1920x1080"
    EndSubSection
EndSection
```

Device option information.

If you have multiple cards that are SLI capable, it is possible to run more than one monitor attached to separate cards (for example: two cards in SLI with one monitor attached to each). The "MetaModes" option in conjunction with SLI Mosaic mode enables this. Below is a configuration which works for the aforementioned example and runs **GNOME** flawlessly.

`/etc/X11/xorg.conf.d/10-monitor.conf`

```
Section "Device"
        Identifier      "Card A"
        Driver          "nvidia"
        BusID           "PCI:1:00:0"
EndSection

Section "Device"
        Identifier      "Card B"
        Driver          "nvidia"
        BusID           "PCI:2:00:0"
EndSection

Section "Monitor"
        Identifier      "Right Monitor"
EndSection

Section "Monitor"
        Identifier      "Left Monitor"
EndSection

Section "Screen"
        Identifier      "Right Screen"
        Device          "Card A"
        Monitor         "Right Monitor"
        DefaultDepth    24
        Option          "SLI" "Mosaic"
        Option          "Stereo" "0"
        Option          "BaseMosaic" "True"
        Option          "MetaModes" "GPU-0.DFP-0: 1920x1200+4480+0, GPU-1.DFP-0:1920x1200+0+0"
        SubSection      "Display"
                        Depth           24
        EndSubSection
EndSection

Section "Screen"
        Identifier      "Left Screen"
        Device          "Card B"
        Monitor         "Left Monitor"
        DefaultDepth    24
        Option          "SLI" "Mosaic"
        Option          "Stereo" "0"
        Option          "BaseMosaic" "True"
        Option          "MetaModes" "GPU-0.DFP-0: 1920x1200+4480+0, GPU-1.DFP-0:1920x1200+0+0"
        SubSection      "Display"
                        Depth           24
        EndSubSection
EndSection

Section "ServerLayout"
```

```
        Identifier      "Default"
        Screen 0        "Right Screen" 0 0
        Option          "Xinerama" "0"
EndSection
```

Vertical sync using TwinView

If you are using TwinView and vertical sync (the "Sync to VBlank" option in **nvidia-settings**), you will notice that only one screen is being properly synced, unless you have two identical monitors. Although **nvidia-settings** does offer an option to change which screen is being synced (the "Sync to this display device" option), this does not always work. A solution is to add the following environment variables at startup, for example append in `/etc/profile`:

```
export __GL_SYNC_TO_VBLANK=1
export __GL_SYNC_DISPLAY_DEVICE=DFP-0
export VDPAU_NVIDIA_SYNC_DISPLAY_DEVICE=DFP-0
```

You can change `DFP-0` with your preferred screen (`DFP-0` is the DVI port and `CRT-0` is the VGA port). You can find the identifier for your display from **nvidia-settings** in the "X Server XVideoSettings" section.

Gaming using TwinView

In case you want to play fullscreen games when using TwinView, you will notice that games recognize the two screens as being one big screen. While this is technically correct (the virtual X screen really is the size of your screens combined), you probably do not want to play on both screens at the same time.

To correct this behavior for SDL, try:

```
export SDL_VIDEO_FULLSCREEN_HEAD=1
```

For OpenGL, add the appropriate Metamodes to your xorg.conf in section `Device` and restart X:

```
Option "Metamodes" "1680x1050,1680x1050; 1280x1024,1280x1024; 1680x1050,NULL; 1280x1024,NULL;"
```

Another method that may either work alone or in conjunction with those mentioned above is **starting games in a separate X server**.

Mosaic mode

Mosaic mode is the only way to use more than 2 monitors across multiple graphics cards with compositing. Your window manager may or may not recognize the distinction between each monitor. Mosaic mode requires a valid SLI configuration. Even if using Base mode without SLI, the GPUs must still be SLI capable/compatible.

Base Mosaic

Base Mosaic mode works on any set of Geforce 8000 series or higher GPUs. It cannot be enabled from within the nvidia-setting GUI. You must either use the `nvidia-xconfig` command line program or edit `xorg.conf` by hand. Metamodes must be specified. The following is an example for four DFPs in a 2x2 configuration, each running at 1920x1024, with two DFPs connected to two cards:

```
$ nvidia-xconfig --base-mosaic --metamodes="GPU-0.DFP-0: 1920x1024+0+0, GPU-0.DFP-1: 1920x1024+1920+0, GPU-1.DFP-0: 1920x1024+0+1024, GPU-1.DFP-1: 1920x1024+1920+1024"
```

Note: While the documentation lists a 2x2 configuration of monitors, **GeForce cards are artificially limited to 3 monitors** in Base Mosaic mode. Quadro cards support more than 3 monitors. As of September 2014, the Windows driver has dropped this artificial restriction, but it remains in the Linux driver.

SLI Mosaic

If you have an SLI configuration and each GPU is a Quadro FX 5800, Quadro Fermi or newer then you can use SLI Mosaic mode. It can be enabled from within the nvidia-settings GUI or from the command line with:

```
$ nvidia-xconfig --sli=Mosaic --metamodes="GPU-0.DFP-0: 1920x1024+0+0, GPU-0.DFP-1: 1920x1024+1920+0, GPU-1.DFP-0: 1920x1024+0+1024, GPU-1.DFP-1: 1920x1024+1920+1024"
```

Wayland

For now only a few **Wayland compositors** support NVIDIA's buffer API, see **Wayland#Requirements** for more information.

For further configuration options, take a look at the wiki pages or documentation of the respective compositor.

Regarding XWayland take a look at **Wayland#XWayland**.

Tips and tricks

See **NVIDIA/Tips and tricks**.

Troubleshooting

See **NVIDIA/Troubleshooting**.

See also

- **Current graphics driver releases in official Nvidia Forum**
- **NVIDIA Developers Forum - Linux Subforum**

Categories: Graphics | X server