Quadro Professional Drivers

Quadro FX 3800/4800/5800 and Quadro CX SDI User's Guide

Version 2.0

Published by
NVIDIA Corporation
2701 San Tomas Expressway
Santa Clara, CA 95050

NVIDIA Corporation
2701 San Tomas Expressway
Santa Clara, CA 95050
www.nvidia.com

Table of Contents

CHAPTER 1

ABOUT NVIDIA GRAPHICS TO SDI

Serial Digital Interface (SDI) is a digital, uncompressed high quality video format used for film and video post production and broadcast applications. The NVIDIA Quadro® CX SDI, NVIDIA Quadro® FX 3800 SDI, NVIDIA Quadro® FX 4800 SDI and NVIDIA Quadro® FX 5800 SDI[1] convert composited video and graphics to uncompressed 8-bit, 10-bit, or 12-bit SDI output.

About This Document

This manual explains the graphics-to-SDI functionality of the NVIDIA Quadro FX SDI graphics card and software, described in the following sections:

- "NVIDIA Graphics-to-SDI" on page 3 lists the supported SDI features and explains the basic operation in a broadcast environment.
- "Windows–Using the Graphics to SDI Control Panel" on page 15 describes how to use the Display Properties control panel to set up and start the SDI output under Windows.
- "Linux—Using the Graphics to Video Out Control Panel" on page 31 describes how to use the Display Properties control panel to set up and start the SDI output under Linux.
- "API Control" on page 47 gives an overview of API control of the SDI functions.

For instructions on installing the graphics card and drivers, refer to the documentation that accompanies your NVIDIA Quadro FX SDI graphics card.

Other Documents

For details on using the NVIDIA Control Panel, see the *NVIDIA Control Panel Quick Start Guide*.

1. In the rest of this document, "NVIDIA Quadro FX SDI" refers to the NVIDIA Quadro CX SDI, Quadro FX 3800 SDI, Quadro FX 4800 SDI, and Quadro FX 5800 SDI products.

System Requirements

- The following operating systems are supported:
  - Windows® XP.
  - Linux
- NVIDIA Quadro CX SDI, NVIDIA Quadro FX 3800 SDI, NVIDIA Quadro FX 4800 SDI, or NVIDIA Quadro FX 5800 SDI Graphics Card
- PCI-Express Motherboard
- NVIDIA Professional Graphics Driver
  - For Windows, version 182.xx or later.
  - For Linux, version 182.00.xx or later.

Revision History

Revision	Date	Description
1.0	10/25/08	Initial Release.
2.0	3/17/09	Added the NVIDIA Quadro FX 3800 SDI

CHAPTER

2

NVIDIA GRAPHICS-TO-SDI

This chapter provides an overview of the NVIDIA graphics-to-SDI functionality, described in the following sections:

- "Feature Overview" on page 4 lists the hardware connections, supported SDI formats, and additional SDI support features of the NVIDIA Quadro FX SDI graphics card.
- "Installing and Preparing the NVIDIA Quadro FX SDI" on page 6 describes how to install the NVIDIA Quadro FX SDI card and prepare it for use.
- "Operating NVIDIA SDI" on page 10 provides an overview of SDI operation.

Feature Overview

Input/Output Connections

- Two BNC connections that can be configured as a single fill + key dual-link SDI output, or up to two fill single-link SDI outputs
- One DVI video monitoring output
- BNC connection for external sync signals

Supported SDI Signal Formats

- Standard Definition (SD) Modes
  - 487i @ 59.95 Hz (SMPTE259) NTSC
  - 576i @ 50.00 Hz (SMPTE259) PAL
- High Definition (HD) Modes
  - 720p @ 23.97 Hz, 24.00 Hz, 25.00 Hz, 29.97 Hz, 30.00 Hz, and 50.00 Hz
  - 720p @ 59.94Hz, 60.00 Hz (SMPTE296)
  - 1035i @ 59.94 Hz, 60.00 Hz (SMPTE260)
  - 1080i @ 50.00 Hz, 59.94 Hz, 60.00 Hz (SMPTE274)
  - 1080PsF @ 24.00 Hz, 23.976 Hz
  - 1080PsF @ 25.00 Hz, 29.97 Hz, 30 Hz (SMPTE274)
  - 1080p @ 23.976 Hz, 24.00 Hz, 25.00 Hz, 29.97 Hz, 30.00 Hz (SMPTE274)
  - 2048x1080p @ 23.976 Hz, 24.00 Hz, 25.00 Hz, 29.97 Hz, 30.00 Hz, 47.96Hz, 48Hz, 60Hz (SMPTE272)

Supported SDI Color Formats

- RGB 4:4:4
- YCrCb 4:2:2 or 4:4:4
- RGBA 4:4:4:4
- YCrCbA 4:2:2:4

Supported Output Modes

- Clone Mode
- Dualview Mode
- Application-controlled Mode using NVIDIA SDI APIs

Desktop Region Adjustment Capability

When in Clone mode, lets you define a portion of the desktop to convert to SDI output.

Genlock and Frame Lock Capability

Lets you synchronize the SDI output to an external digital or analog sync source.

Note: The NVIDIA Quadro FX SDI card does not support SLI mode at this time.

Installing and Preparing the NVIDIA Quadro FX SDI

About Your NVIDIA Quadro FX SDI

The following describes the components included in your NVIDIA Quadro FX SDI product package:

Cards

The NVIDIA Quadro FX SDI consists of the following two cards:

- NVIDIA Quadro CX, NVIDIA Quadro FX 3800, Quadro FX 4800, or Quadro FX 5800 graphics card
- NVIDIA SDI Output Card

Cables

In addition, you need the following cables, which should be provided with your NVIDIA Quadro FX SDI package:

- (Qty 1 ea.) 14-Pin Ribbon Cable

  This cable connects the NVIDIA Quadro CX / FX 3800/4800/5800 card to the SDI Output card for genlock and frame-lock functionality.

- (Qty 1 ea.) DVI-to-DVI Cable

  This cable connects the video output from the graphics card to the SDI output card.

Installing the NVIDIA Quadro FX SDI

Step 1: Install the NVIDIA Quadro FX SDI

1 Power down the system and open the chassis cover.

2 Install the NVIDIA Quadro FX card

a Insert the graphics card into the x16 PCI-express slot and use a screw to secure the card's bracket to the system chassis.

b Connect the power cable to the auxiliary power connector(s).

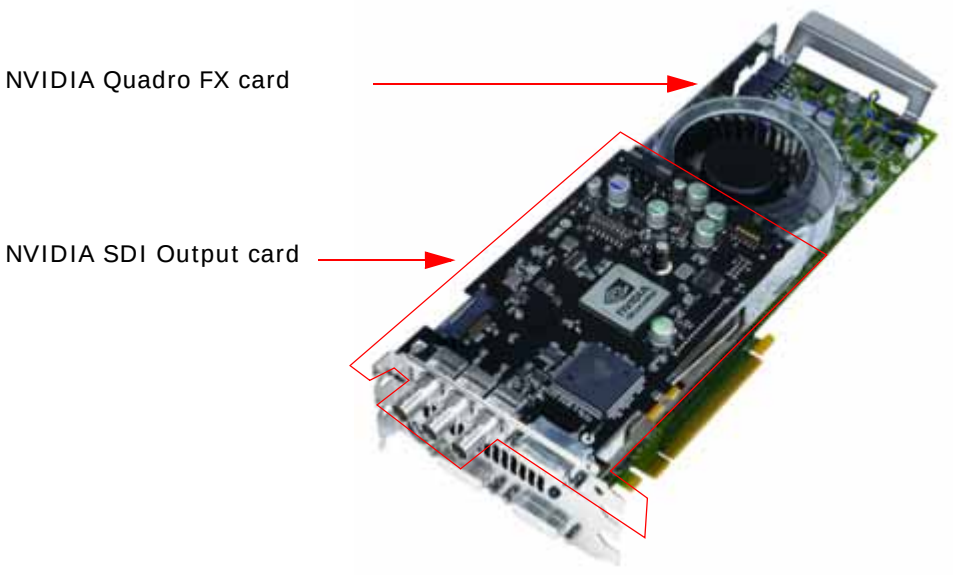
The NVIDIA Quadro FX 5800 requires power to two auxiliary power connections.

3 Install the NVIDIA SDI Output card.

a Insert the NVIDIA SDI Output card into any available type of expansion slot within six inches of the NVIDIA Quadro FX G-Sync connector, and use a screw to secure the card's bracket to the system chassis.

b Connect the power cable to the auxiliary power connector.

4 Connect one end of the 14-pin ribbon cable to the G-Sync connector on the NVIDIA Quadro FX card, and the other end to the NVIDIA SDI Output card.



5 Close the chassis cover.

Step 2: Connect the Auxiliary Cabling and Monitor

1 Connect the DVI Connectors.

Connect one end of the DVI cable to the DVI connector on the SDI Output card, and the other end to the "north" DVI connector on the NVIDIA Quadro FX SDI card as shown in Figure 2.1, Figure 2.2, and Figure 2.3.

The cable must be connected to the "north" DVI connector. The NVIDIA Quadro FX SDI will ***not*** work properly if the cable is connected to the other digital connectors. .

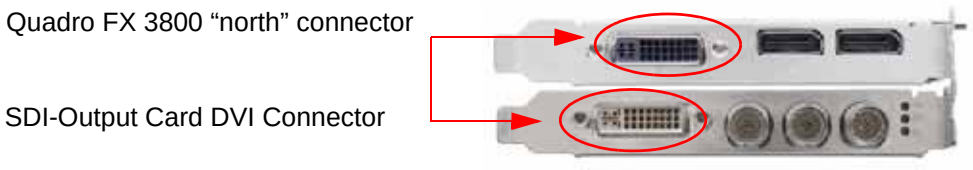


Figure 2.1 DVI Connection between the SDI Output card and the Quadro FX 3800

.

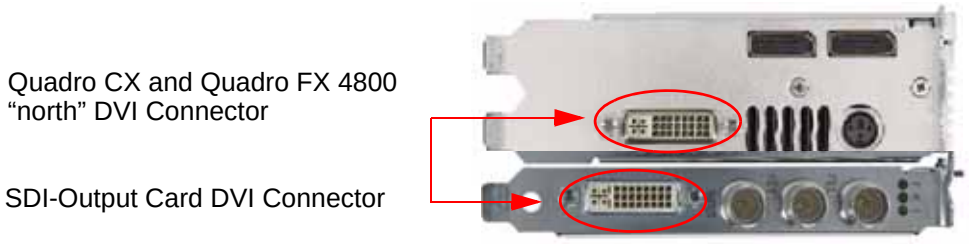


Figure 2.2 DVI Connection between the SDI Output card and the Quadro CX/QuadroFX 4800

.

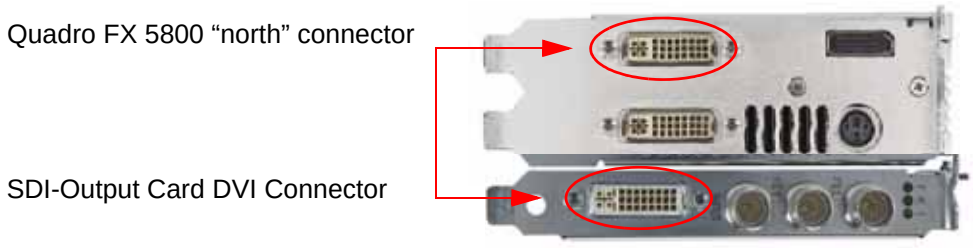


Figure 2.3 DVI Connection between the SDI Output card and the Quadro FX 5800

2 Connect your display to one of the available digital connectors on the graphics card as shown in Figure 2.4.

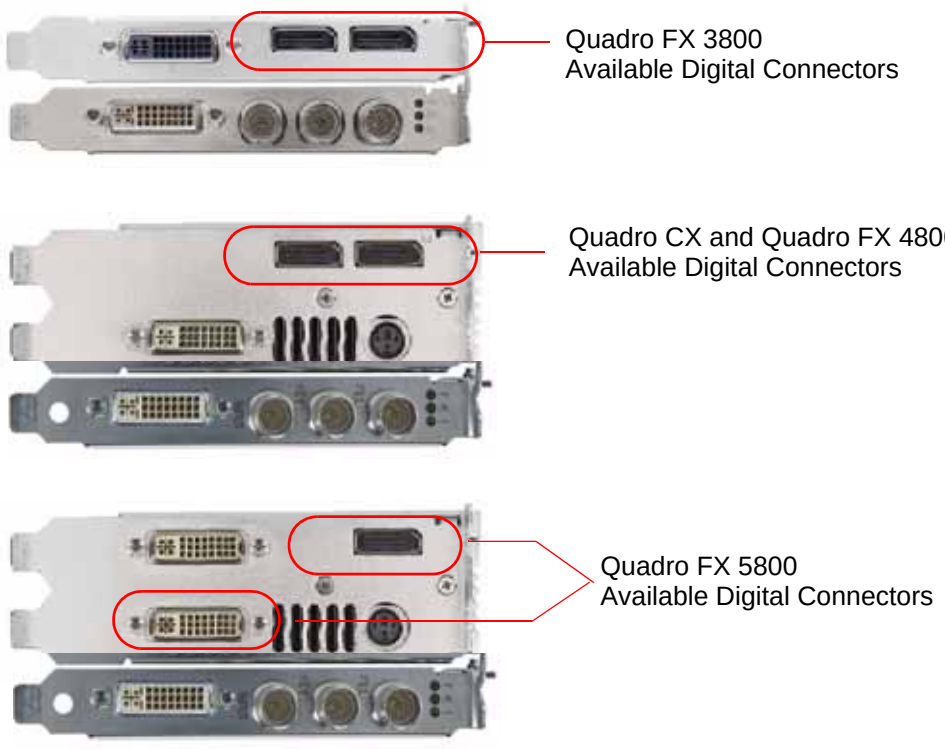


Figure 2.4 Digital Connectors Available for Displays

Step 3: Install the NVIDIA Graphics Drivers

If you will be installing new graphics drivers for the NVIDIA Quadro FX SDI card, it is highly recommended that you uninstall any previous version of the NVIDIA ForceWare graphics driver software before installing updated graphics drivers.

1 Follow the instructions on the NVIDIA.com Web site driver download page to locate the appropriate driver to download, based on your hardware and operating system.

2 Click the driver download link.

The license agreement dialog box appears.

3 Click **Accept** if you accept the terms of the agreement, then either open the file or save the file to your PC and open it later.

Opening the EXE file launches the NVIDIA InstallShield Wizard.

4 Follow the instructions in the NVIDIA InstallShield Wizard to complete the installation.

Operating NVIDIA SDI

The following sections provide an overview of SDI operation:

Understanding the Connections

Figure 2.5 shows the available SDI and external sync connectors on the NVIDIA Quadro FX SDI.

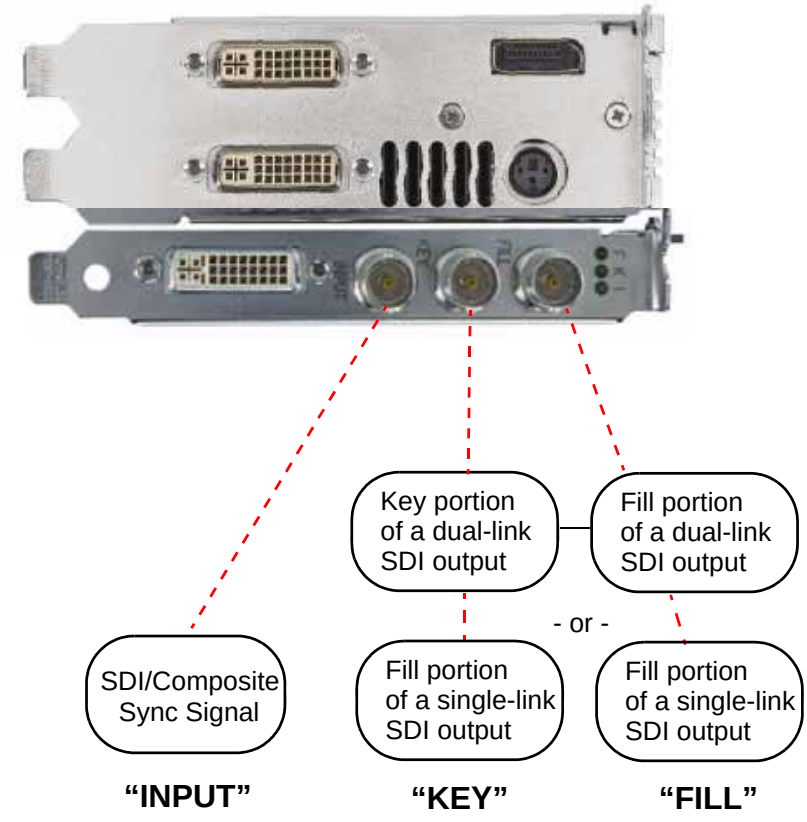


Figure 2.5 NVIDIA Quadro FX SDI Connectors

Connecting the SDI Video Output

Refer to Figure 2.5.

- 4:4:4/4:2:2:4/4:4:4:4 dual-link signals are sent to the **FILL** and **KEY** connectors.
- 4:2:2 single-link signals are sent to the **FILL** connector only.

  In application control mode, using the APIs, an additional 4:2:2 signal can be sent to the **KEY** connector.

Connecting to an External Sync Source

- You can genlock the output to an external digital or analog sync source. Several systems can also be frame-locked.

  NVIDIA Genlock supports the following two external synchronization signal types:

  - SDI
  - Composite, which can be one of the following:

    Composite Bi-level (NTSC or PAL sources use bi-level composite signals.)

    Composite Tri-level (HDTV sources commonly use tri-level composite signals.)
- To use an external sync source, connect the sync signal to the INPUT BNC connector as indicated in Figure 2.5, then select the corresponding signal type (SDI or composite) using the NVIDIA Control Panel.

About the Software

The NVIDIA SDI software lets you specify the

- SDI signal format
- Color formats
- Synchronization method
- Gamma correction
- Color-space conversion

Graphics-to-SDI functionality can be set up and controlled in two basic ways—using the NVIDIA Control Panel for 8-bit SDI output from the desktop, or using the NVIDIA SDI API for 8-, 10-, or 12-bit SDI output from an application.

Using the SDI APIs

The SDI application programming interface allows OpenGL applications to have full and exclusive control of the SDI output.

When the SDI output is under application control, you can view the SDI hardware status using the NVIDIA Control Panel **Send graphics to SDI ouput** page.

- See the chapter "API Control" on page 47 for a description of the graphics-to-video-out API calls.
- Also, refer to the document *Programming the NVIDIA Quadro FX 4800/5800 SDI* for instructions on using the APIs.

Using the Control Panel

When the SDI output is ***not*** being controlled by an application, the SDI software works on top of existing applications, and the active workstation desktop or full screen application display is automatically forwarded to the SDI video outputs. This is accomplished under either Clone or Dualview mode.

In this mode, you can use the NVIDIA **Graphics to SDI** property page to

- Configure the external synchronization signal if needed.
- Specify the SDI signal format, output format, and then enable the SDI output.

For detailed instructions under Windows, see the chapter "Windows–Using the Graphics to SDI Control Panel" on page 15.

For detailed instructions under Linux, see the chapter "Linux—Using the Graphics to Video Out Control Panel" on page 31.

Recommended Operating Practices

This section provides some basic operating practices to follow in order to obtain the best SDI performance for your application.

Initial On-Air Broadcast

When starting a live broadcast of SDI video, follow the sequence below to ensure proper allocation of system resources and to prevent visual disturbances in the on air broadcast.

1. Set up the SDI format settings and start the SDI output
2. Start the application to be broadcast
3. Verify the video quality
4. Close the Graphics to SDI control panel
5. Go on air

   To avoid visual disturbances while broadcasting live, DO NOT

   - Start or stop the graphics or video application
   - Turn on or off the SDI output
   - Make changes to the SDI signal format

Changing Applications

To avoid visual disturbances while switching applications, observe the following sequence:

1. Stop the live broadcast (go off air)
2. Stop the application
3. Start the new application
4. Verify video quality
5. Resume the live broadcast

Changing Video Formats

When changing any of the SDI settings, visual disturbances might occur as the video resets to the new settings. To prevent such disturbances from being visible to the public or from being recorded, observe the following sequence when making changes to any SDI setting:

1. Stop the live broadcast (go off air)
2. Change video format or SDI settings
3. Verify video quality
4. Resume the live broadcast

When Using the Control Panel

NVIDIA recommends the following

- Set the desktop to the same or higher resolution than the SDI output for better image quality.
- Close all background applications—such as virus scan, backup, and archiving applications—before starting the SDI output and going on air.
- Close the Display Properties panel before going on air.
- When running multiple OpenGL applications, tearing may occur if the applications are not synchronized.

  In general, NVIDIA does not recommend running multiple OpenGL applications when starting the SDI output or when going live.

Running Multiple OpenGL Applications

To maximize the system resources and bandwidth available for converting graphics to SDI output, NVIDIA recommends broadcasting only one OpenGL application at a time.

CHAPTER

3

WINDOWS–USING THE GRAPHICS TO SDI CONTROL PANEL

This chapter explains how to set up the NVIDIA Quadro FX SDI graphics card for Windows under Clone or Dualview mode using the NVIDIA Control Panel **Send Graphics to SDI output** page. It contains the following sections:

How to Set Up the Graphics-to-SDI Output

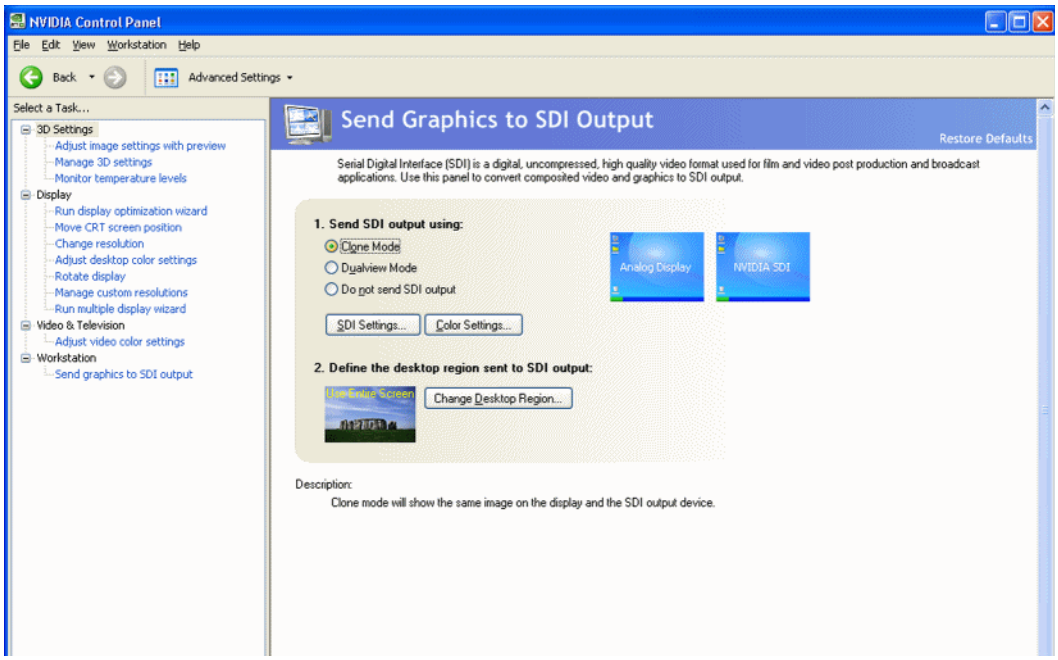
This section explains how to set up the graphics-to-SDI output.

Basic SDI Setup

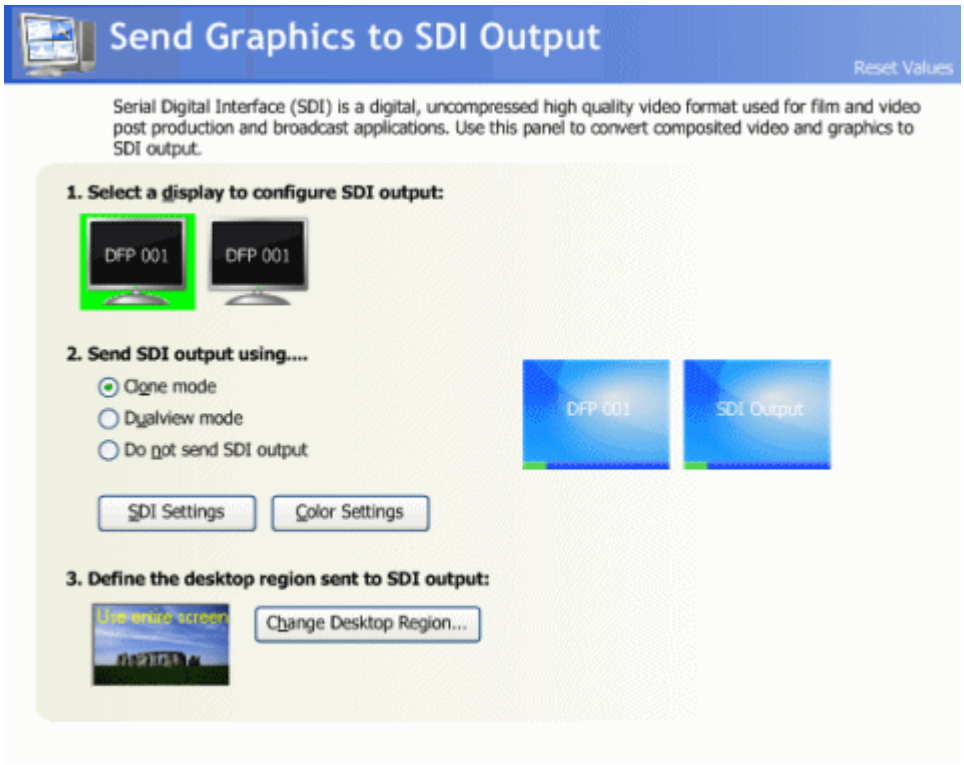
To ensure proper operation, NVIDIA recommends the following -

- ***Set the desktop resolution to be the same or larger than the SDI output for better image quality***
- ***Stop background applications—such as virus scan, backup and archiving applications—prior to starting SDI output and going on air.***
- ***Close the control panel before going on air.***
- ***When running multiple OpenGL applications, synchronize them, otherwise tearing may occur.***

1 From the NVIDIA Control Panel navigation tree pane, under Workstation, click **Send graphics to SDI output**.



2 If you are using more than one NVIDIA Quadro FX SDI card, under **Select a display to configure SDI output** click the display icon corresponding to the display you want to configure, then follow the remaining instructions for that display.

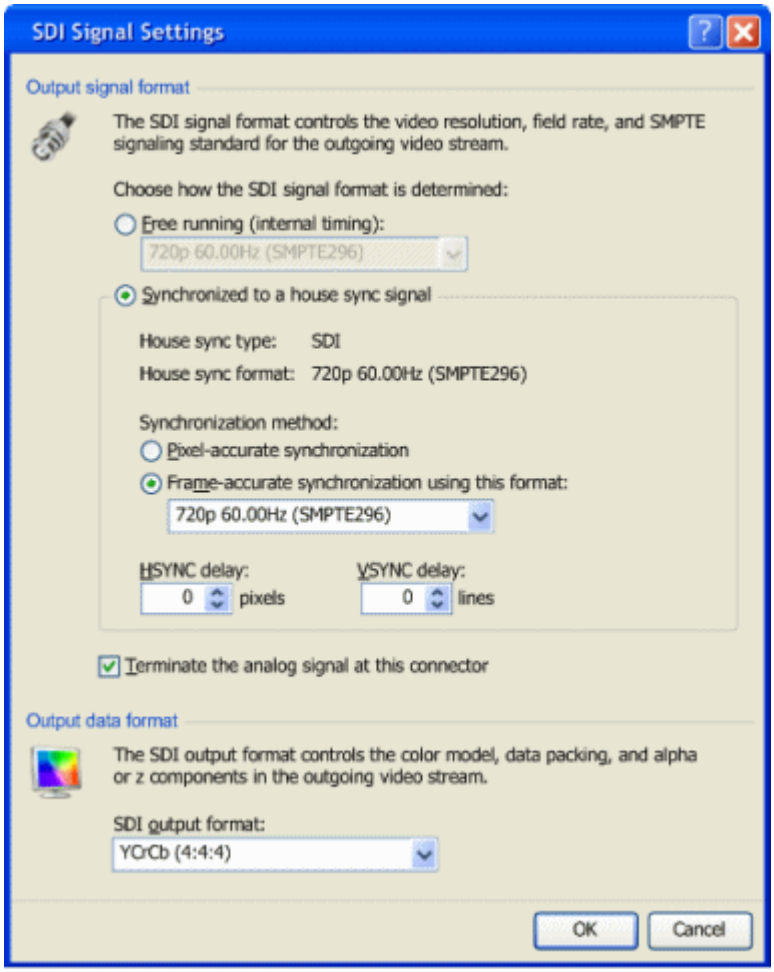


- This option does not appear if your system contains only one NVIDIA Quadro FX SDI card.
- If this option does not appear and your system does contain more than one NVIDIA Quadro FX SDI card, see "Enabling Multiple SDI Cards" on page 27 for instructions on enabling the cards.

3 Under **Send SDI output using**, select the SDI output mode that you want to use.

- **Clone mode**: In Clone mode, the SDI output is a clone of the display output.
- **Dualview mode**: In Dualview mode, you can define one large desktop that extends from the display to the SDI output. This lets you move windows between the SDI output and the graphics (DVI) display part of the extended desktop.
- **Do not send SDI output**: With this option, no signal is sent to the SDI output. The remaining controls on the page are disabled. Choose this option if you want an application to control the SDI output. Once the application is running, this page does not let you change the settings, but only shows the settings established by the application.

4 Click the SDI Settings bar to open the SDI Signal Settings dialog box.



5 Choose a method for determining the format of the SDI output - either using internal timing or synchronized to an external signal source.

- To use internal timing, select **Free running (internal timing)**, then click the list arrow and choose from the list of available SDI signal formats.
- To synchronize to an external signal source, make sure the house sync is connected to the INPUT BNC connector on the graphics card, then select the **Synchronized to a house sync signal** radio buttion and set up the synchronization and signal formats as follows:
  - Select the **House sync type** radio button (SDI or Composite) that corresponds to the sync signal type you are using.
  - To synchronize the pixel scanning of the SDI output to the external signal using genlock, select **Pixel-accurate synchronization**.

- To synchronize the frame rate of the SDI output to the external signal using frame lock, select **Frame-accurate synchronization using this format**, then click the list arrow and choose from the list of available SDI signal formats.
- To introduce a delay in the SDI output, enter the pixel or line delay values in the appropriate HSYNC or VSYNC delay boxes.

See "Synchronizing the SDI Output to an External Source" on page 20 for additional information.

6. Check the Terminate the analog signal at this connector check box if
   - the system is a standalone system synchronized to a house sync signal, or
   - the system is the last in a chain of systems connected to the same house sync signal.
7. Choose the SDI output data format by clicking the SDI output format list arrow and then selecting from the list of available color formats.
8. Click **OK** when done to close the dialog box.
9. Click **Apply**.

Synchronizing the SDI Output to an External Source

You can synchronize the SDI output with other equipment in a broadcast or post production environment.

Supported Synchronization Methods

The **Graphics to SDI** page provides two methods for synchronizing the SDI output to a common sync source—pixel-accurate or frame-accurate synchronization.

- **Pixel-accurate synchronization** synchronizes the pixel scanning of the SDI output to the house sync signal. When using pixel-accurate synchronization, the SDI refresh rate is determined by the sync signal.
- **Frame-accurate synchronization** synchronizes the frame rate of the SDI output to the house sync signal. The sync signal determines the available SDI signal formats.

Supported Synchronization Signals

The NVIDIA driver supports the following external synchronization signal types:

- **SDI**
- **Composite Bi-level** (NTSC or PAL sources use bi-level composite signals.)
- **Composite Tri-level** (HDTV sources commonly use tri-level composite signals.)

Connecting to an External Synchronization Source

To use an external sync source:

1. Connect the sync signal to the INPUT BNC connector.

   You can connect multiple systems to the same house sync by daisy-chaining the house sync cable to each card using BNC T-connectors.

2. Follow the instructions in Basic SDI Setup for setting up your SDI system to use the external sync signal.

   The driver will not detect a valid sync signal until the correct signal type is configured in the NVIDIA Control Panel.

Understanding the Status Indicators

The LEDs on the NVIDIA SDI Output Card connector bracket indicate the status of the SDI outputs and the synchronization input signals.

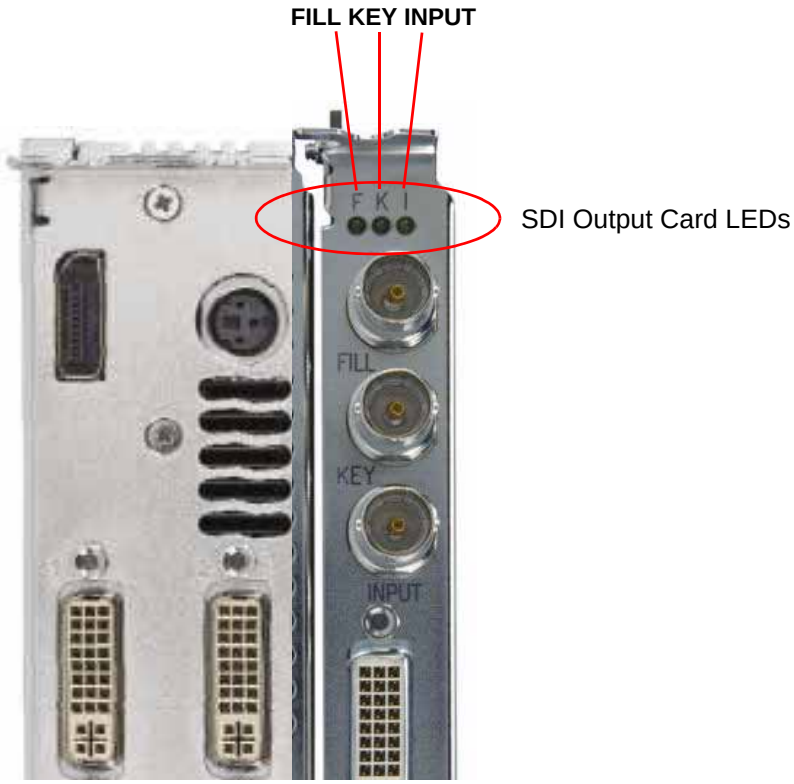


Figure 3.1 SDI Connection LED Indicators

The activity of the LEDs indicates the signal status as follows:

- **FILL** or **KEY Out**

Status	Meaning
Off (gray)	SDI output is not in use
Steady Green	SDI output has power.
Blinking Green	SDI output is active.

- **Input**

Status	Meaning
Off (gray)	SDI input synchronization is disabled.
Blinking Green	Valid SDI synchronization signal is detected.

Advanced Adjustments

This section describes the following additional settings that you can control using the Graphics to SDI Output page:

- "Adjusting the Desktop Area" on page 22
- "Applying Gamma Correction" on page 24
- "Setting Up the Color Space Conversion" on page 25
- "Synchronizing the SDI Output to an External Source" on page 20

Adjusting the Desktop Area

By default, the entire desktop is converted to SDI output. If the desktop is smaller than the size of the SDI output, it will be scaled to fit. If the desktop is larger than the SDI output, it will be cropped to fit.

Instead of using the entire desktop, you can specify a region of the desktop to convert to SDI output as follows:

1. From the NVIDIA Control Panel navigation tree pane, under Workstation, click Send graphics to SDI output.
2. Click the Change Desktop Region bar.

The NVIDIA Control Panel minimizes and the SDI Output dialog box appears. Superimposed over the desktop is a rectangular outline that shows the region that will be used for the SDI output.

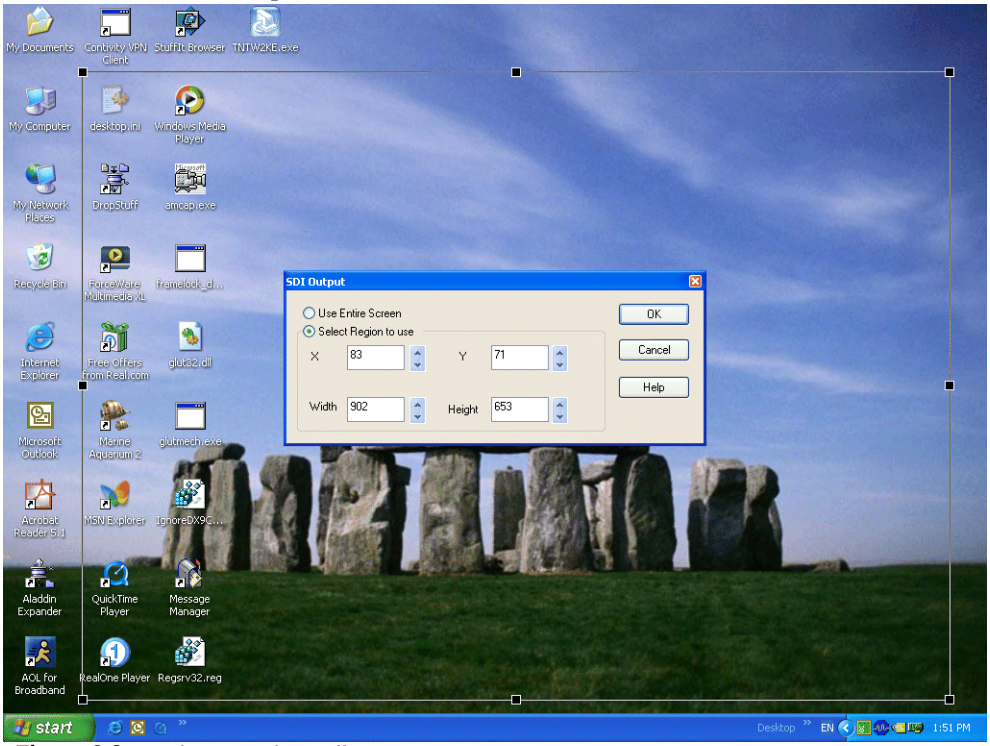


Figure 3.2 Desktop Region Adjustment

3 Click the **Select Region** to use option.

4 Adjust the region size.

- Click and drag within the rectangular outline to adjust its position on the desktop.
- Click and drag the appropriate corner or side handles to resize the outline.
- You can also adjust the region size by specifying the **X**, **Y**, **Width**, and **Height** values in the SDI Output dialog box.

  Either enter pixel values directly into the corresponding text boxes or click the up and down arrows by the appropriate box.

Note: The **X** and **Y** values indicate the distance, in pixels, between the upper-left corner of the desktop and the upper-left corner of the output box.

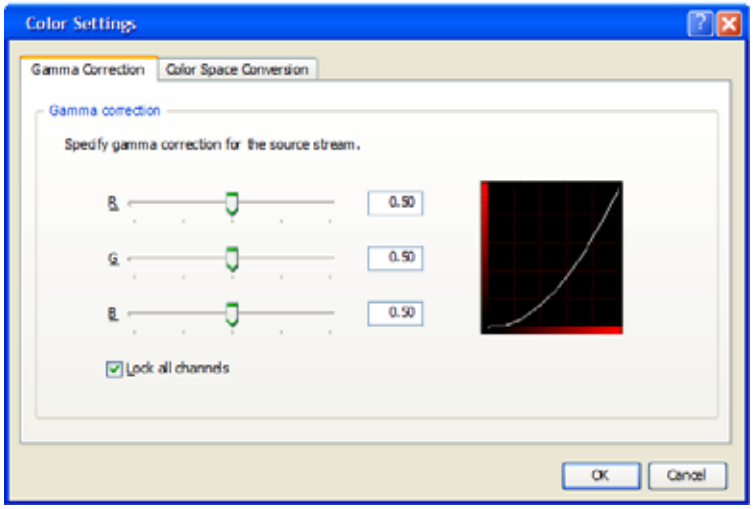
5 Click **OK** when finished.

The desktop graphic image shows a thumbnail preview of the desktop region that you have set up for SDI output.

Applying Gamma Correction

To specify the gamma correction to use for the source stream:

1. From the NVIDIA Control Panel navigation tree pane, under Workstation, click Send graphics to SDI output.
2. Click the Color Settings bar to open the Color Settings dialog box.
3. Click the Gamma Correction tab.

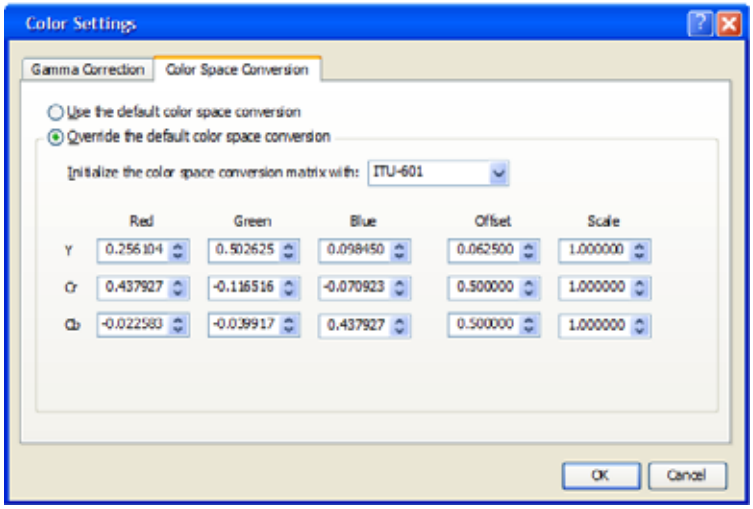


4. Specify the RGB Gamma values using one or more of the following methods:
   - Click and drag each R, G, or B slider to the appropriate value.
   - Enter the R, G, or B value in the respective boxes or use the up and down arrows.

   To keep all gamma channels at the same value while you adjust them simultaneously, click the Lock all channels check box.
5. Click **OK** when finished.

Setting Up the Color Space Conversion

1. From the NVIDIA Control Panel navigation tree pane, under Workstation, click Send graphics to SDI output.
2. Click the Color Settings button to open the Color Settings dialog box.
3. Click the Color Space Conversion tab.



4. Check Override the default color space conversion.
5. Click the Initialize the color space conversion matrix with list arrow and then click one of the pre-defined color-space standards to use as a starting point.
6. In each color-space text box, either enter values directly or use the corresponding up and down arrows to change the values.
7. Click **OK** when finished.

About Dualview Mode

In the default configuration, the SDI output is a clone of the display output. The NVIDIA Quadro FX SDI graphics cards also supports Dualview mode, where the desktop extends across two monitors.

Under Dualview mode, you can define one large desktop that extends from the display to the SDI output. This lets you move windows between the SDI output and the graphics (DVI) display part of the extended desktop.

With applications that use video overlay or Microsoft VMR, you can also display the video full-screen on the SDI output.

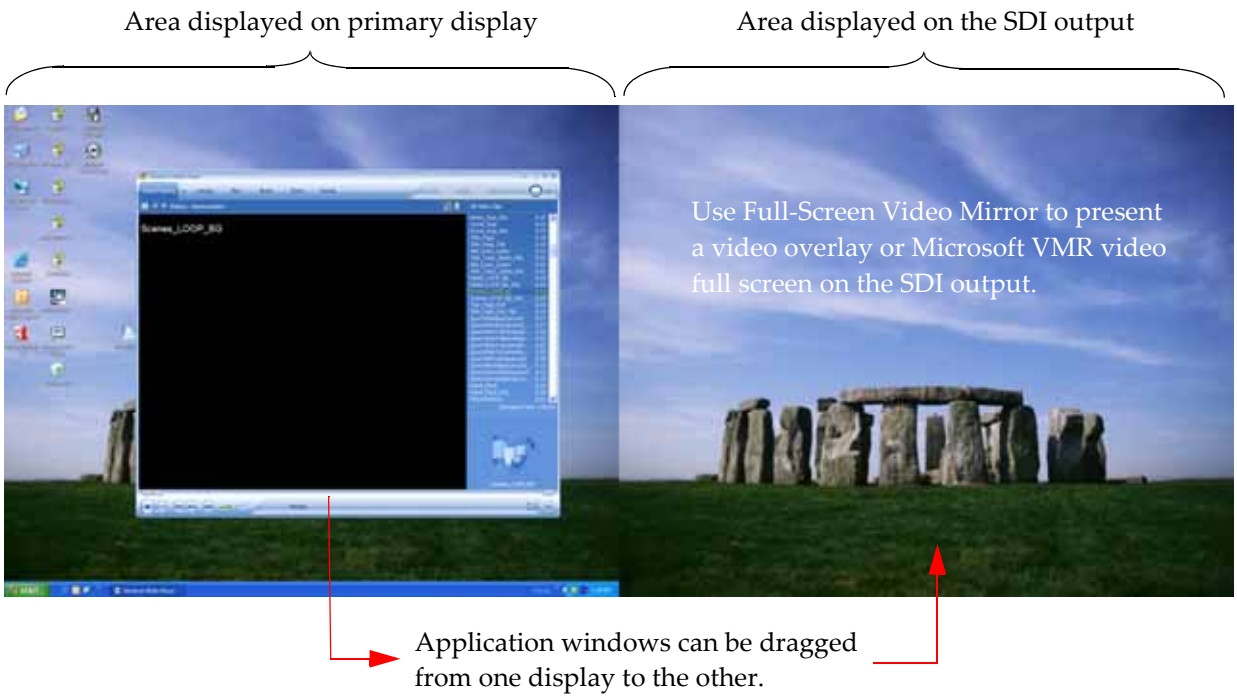


Figure 3.3 Extended Desktop with Dualview Mode

The display and the SDI output do not need to be the same resolution and refresh rate.

Enabling Multiple SDI Cards

On systems with more than one NVIDIA Quadro FX SDI card, the Send Graphics to SDI Output page lets you configure the SDI output for each card. Before you can do this, all cards must be enabled.

To enable multiple SDI cards

1. Make sure a display is connected to each SDI card that you want to enable.
2. Open the Windows Display Properties page.
   a. Right-click the desktop, then click Properties from the pop-up menu.
   b. Click the Settings tab.
3. Determine which monitor icon corresponds to the graphics card that you want to enable.

   There should be two monitor icons for each graphics card in the system. Typically, monitors 1 and 3 are connected to one graphics card and monitors 2 and 4 are connected to the other. For example, if monitor 1 is already attached, then monitor 2 would be grayed out, indicating that it is connected to the graphics card that is not yet enabled.
4. Right-click the grayed-out monitor icon corresponding to the graphics card you want to enable, then click Attached from the pop-up menu.

   - or -

   Click the monitor icon, then click the Extend my Windows desktop onto this monitor check box.
5. Click **OK**.

You can now configure the SDI output for each SDI card as described in Basic SDI Setup.

Open the View System Topology page (see "Viewing the SDI Connection Status Using the Topology Viewer" on page 30) to verify your display-to-graphics card connections.

Allowing Application Control of the SDI Output

The SDI application programming interface allows OpenGL applications to have full and exclusive control of the SDI output.

Refer to the document *Programming the NVIDIA Quadro FX 4800/5800 SDI* for instructions on using the APIs.

To allow applications to control the SDI output -

Step 1: Turn off NVIDIA Control Panel SDI output control.

1. From the NVIDIA Control Panel navigation tree pane, under Workstation, click **Send graphics to SDI output**.
2. If you are using more than one NVIDIA Quadro FX SDI card, under *Select a display to configure SDI output*, click the display icon corresponding to the display you want to configure.
   - This option does not appear if your system contains only one NVIDIA Quadro FX SDI card.
   - If this option does not appear and your system does contain more than one NVIDIA Quadro FX SDI card, see "Enabling Multiple SDI Cards" on page 27 for instructions on enabling the cards.
3. Under **Send SDI output using**, select **Do not send SDI output**.

Step 2: Close the NVIDIA Control Panel.

Step 3: Start the application.

Once the application is running and the SDI output is under application control, you can view the SDI settings and check the status using the **Send Graphics to SDI Output** page.

To view the SDI status, open the NVIDIA Control Panel and click **Send graphics to SDI output** from the *Select a Task* pane.

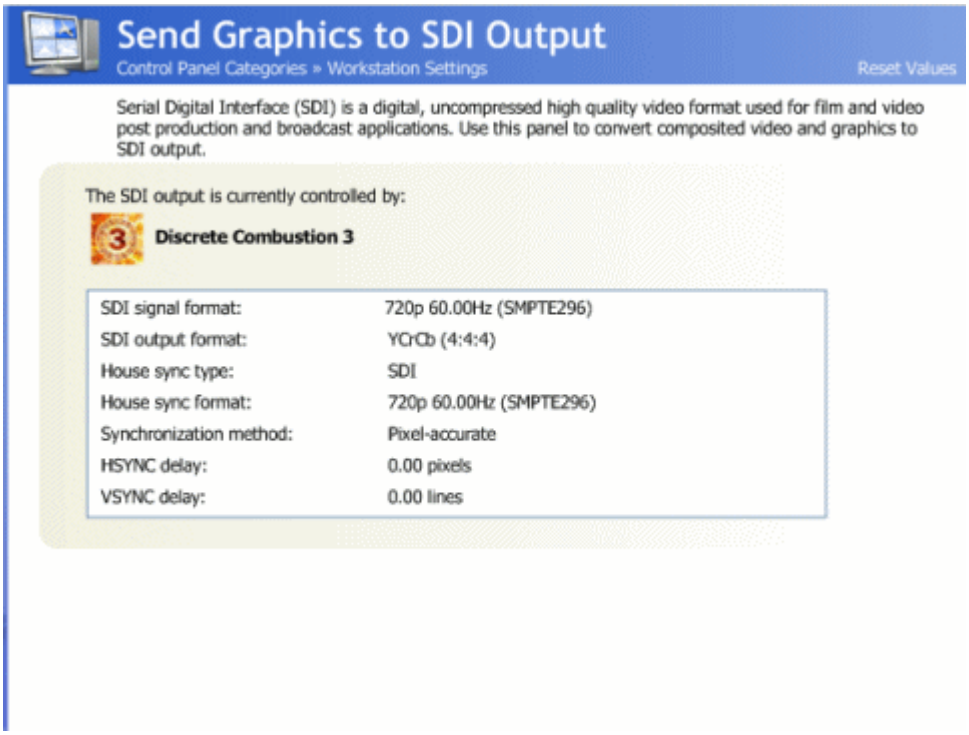


Figure 3.4 Graphics to SDI Page—Application Control

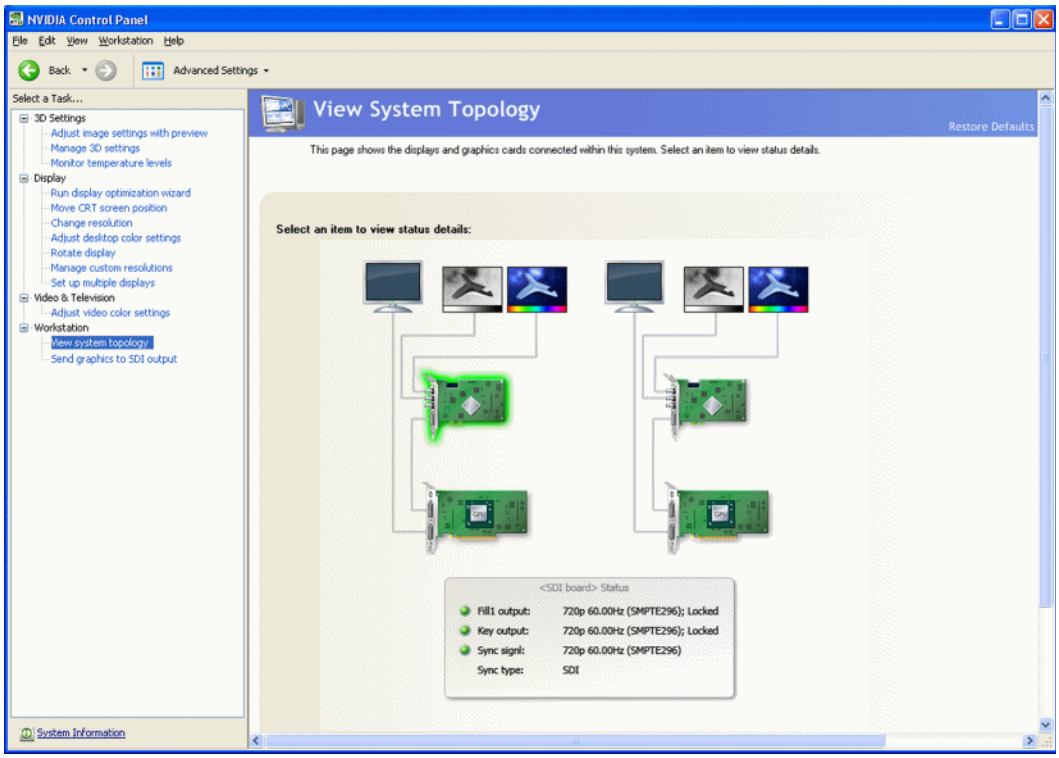
Viewing the SDI Connection Status Using the Topology Viewer

For workstation systems, a graphical topological view of the system is available to let you quickly check the status of your particular graphics environment.

The View System Topology page provides SDI status information for each display, the graphics card-to-SDI card pairing, and the connection status information for the NVIDIA Quadro FX SDI cards. In addition to viewing status information, you can also change various settings using the View System Topology page.

To view the system topology for your graphics-to-SDI setup,

1 From the NVIDIA Control Panel *Select a Task* pane, under Workstation, click **View system topology**.



2 Click any of the icons to view connection and signal status details.

3 You can also right-click the SDI output card icon to access the context menu where you can open the SDI signal settings or color settings dialog boxes.

CHAPTER

4

LINUX—USING THE GRAPHICS TO VIDEO OUT CONTROL PANEL

This chapter explains how to set up the NVIDIA Quadro FX SDI graphics cards under Linux using the NVIDIA **Graphics to Video Out** properties page[1].

It contains the following sections:

- "How to Set Up the SDI Output" on page 32 provides step-by-step instructions for using the control panel to set up the SDI output.
- "Advanced Setups" on page 41 explains other controls that are available besides the basic setup controls.

1. This method of controlling the SDI output is also known as 'transparent mode'.

How to Set Up the SDI Output

This section describes how to set up SDI output on the linux system. There are four methods of using the SDI output. Each are mutually exclusive–you cannot use the SDI output in more than one mode at a time.

- **Clone mode**: In Clone mode, the SDI output is a clone of the display output. This is the default mode. You can switch directly to Dualview/Twinview mode while operating the SDI output.

  See "Basic SDI Setup Under Clone Mode" on page 32.

- **Dualview mode (TwinView)**: In Dualview mode the SDI device is treated as a regular flat panel and you can define one large desktop that extends from the display to the SDI output. This lets you move windows between the SDI output and the graphics (DVI) display part of the extended desktop.

  See "Basic SDI Setup with X-window or under Dualview Mode" on page 36.

- **X-screen mode**: You can display the SDI output on an x-window. In X-screen mode the SDI device is treated as a flat panel that gets its own X screen.

  See "Basic SDI Setup with X-window or under Dualview Mode" on page 36.

- **OpenGL application control**: The SDI application programming interface allows OpenGL applications to have full and exclusive control of the SDI output.

  To use this mode, run an application that uses either SDI APIs to make use of the SDI device.

Basic SDI Setup Under Clone Mode

To ensure proper operation, NVIDIA recommends the following -

- ***Set the desktop resolution to be the same or larger than the SDI output for better image quality***
- ***Stop background applications—such as virus scan, backup and archiving applications—prior to starting SDI output and going on air.***
- ***Close the control panel before going on air.***
- ***When running multiple OpenGL applications, synchronize them, otherwise tearing may occur.***

Step 1: Open the NVIDIA Graphics to Video Out Property Page

1 From the command line, enter "**`nvidia-settings`**"

The NVIDIA X Server Settings page appears.

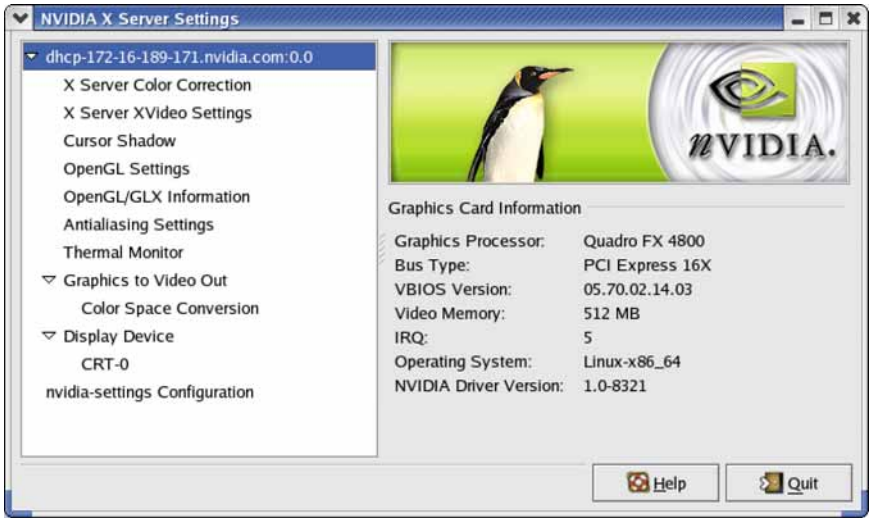


Figure 4.1 NVIDIA X Server Settings Page

2 Click the **Graphics to Video Out** tree item from the side menu.

The **Graphics to Video Out** page appears.

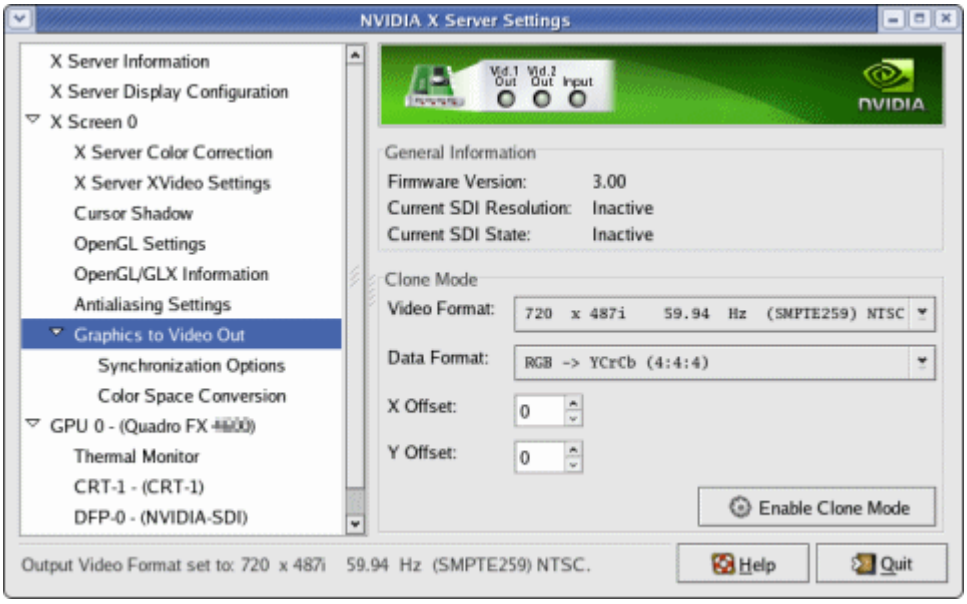
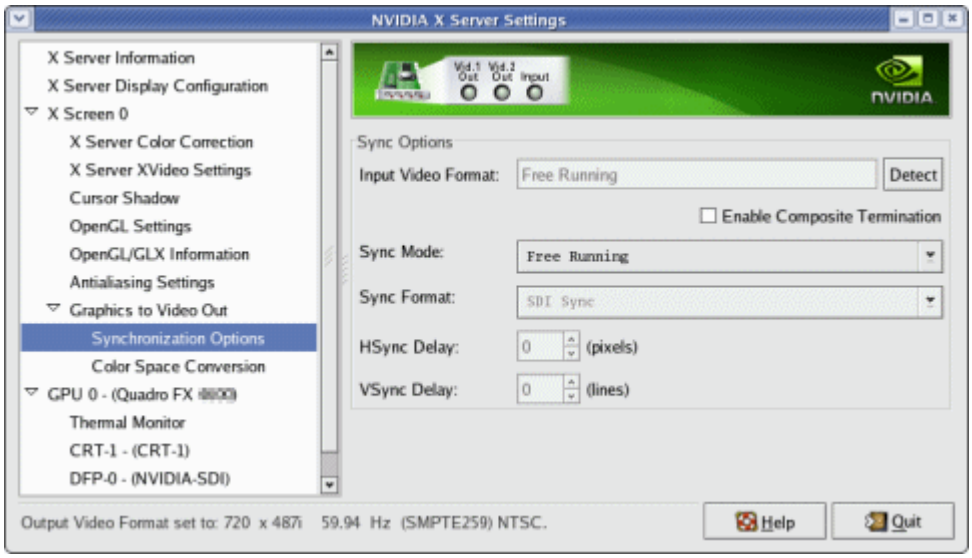


Figure 4.2 Graphics to Video Out Page

Step 2: Choose a Synchronization Method

1 Click the **Graphics to Video Out: Synchronization Options** tree item from the side menu.

The Sync Options page appears.



2 From the **Sync Options** group box, click the **Sync Mode** list arrow and then click the method you want to use to synchronize the SDI output:

- **Free Running:** The SDI output will be synchronized with the timing chosen from the SDI signal format list.
- **Genlock:** The SDI output will be synchronized with the external sync signal.
- **Frame Lock:** The SDI output will be synchronized with the timing chosen from the SDI signal format list.

This list is limited to timings that can be synchronized with the detected external sync signal.

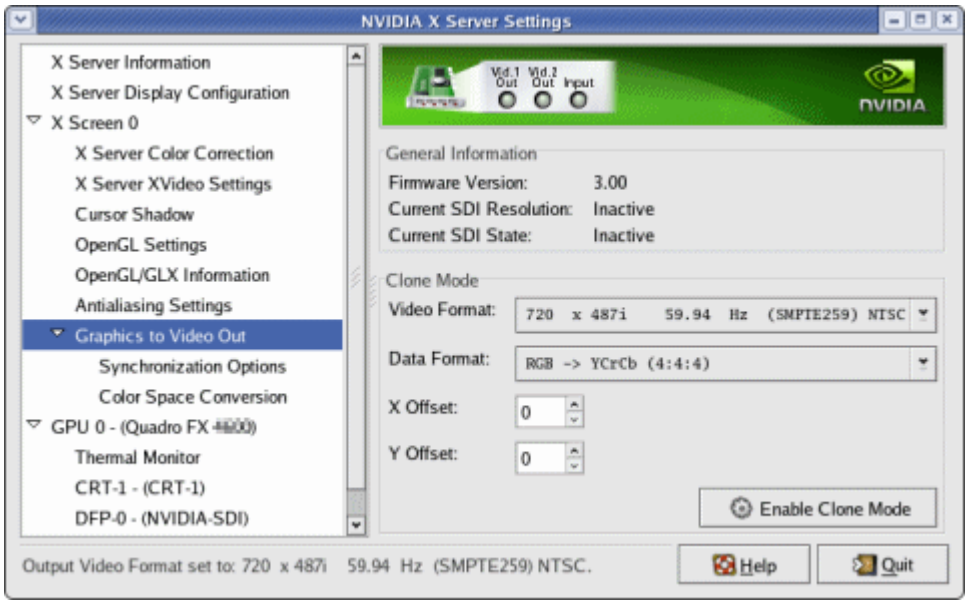
3 Check the **Enable Composite Termination** check box if

- the system is a standalone system synchronized to a house sync signal, or
- the system is the last in a chain of systems connected to the same house sync signal.

For more information regarding genlock and frame lock, see the section "Synchronizing the SDI Output to an External Source" on page 44.

Step 3: Choose the Video and Data Formats

1 Click the **Graphics to Video Out** tree item from the side menu.



2 Specify the video format.

Click the **Video Format** arrow and then click the signal format you want to use.

Video Format controls the video resolution, field rate, and SMPTE signalling standard for the outgoing video stream.

Note: Only those resolutions that your monitor supports appear in the Video Format list. Your options for this setting also depend on which Sync option you chose in the previous step.

- If you chose *genlock synchronization*, the sync source controls the output video format. The list box will be grayed out, preventing you from choosing another format.
- If you chose *frame lock synchronization*, only those modes that are compatible with the detected sync signal will appear in the Output Video Format list.

3 Specify the Data Format

Click the **Output Data Format** arrow and then click the color format you want to use.

Data Format controls the color model, data packing, and alpha or z components in the outgoing video stream.

Step 4: Begin SDI Output

Click **Enable Clone Mode**.

Basic SDI Setup with X-window or under Dualview Mode

To ensure proper operation, NVIDIA recommends the following -

- ***Set the desktop resolution to be the same or larger than the SDI output for better image quality***
- ***Stop background applications—such as virus scan, backup and archiving applications—prior to starting SDI output and going on air.***
- ***Close the control panel before going on air.***
- ***When running multiple OpenGL applications, synchronize them, otherwise tearing may occur.***

Step 1: Configure the Display for Dualview or X-Screen

1 From the command line, enter "**`nvidia-settings`**"

The NVIDIA X Server Settings page appears.

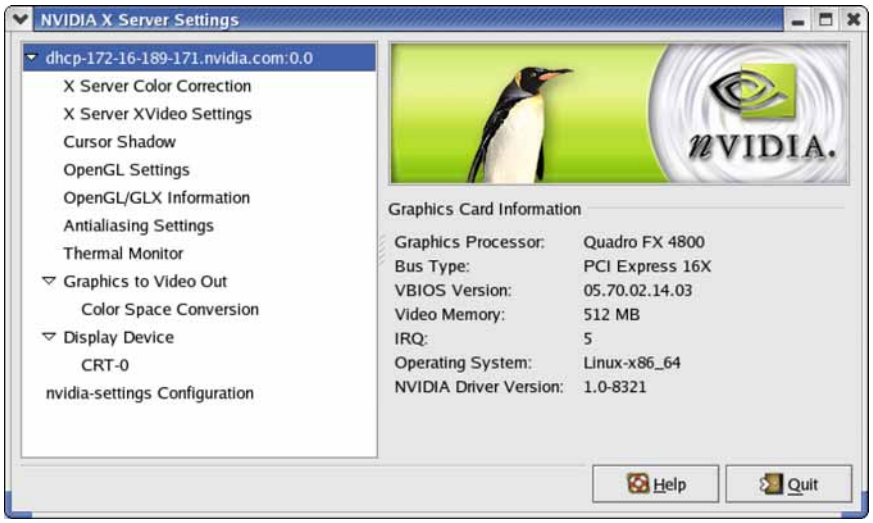
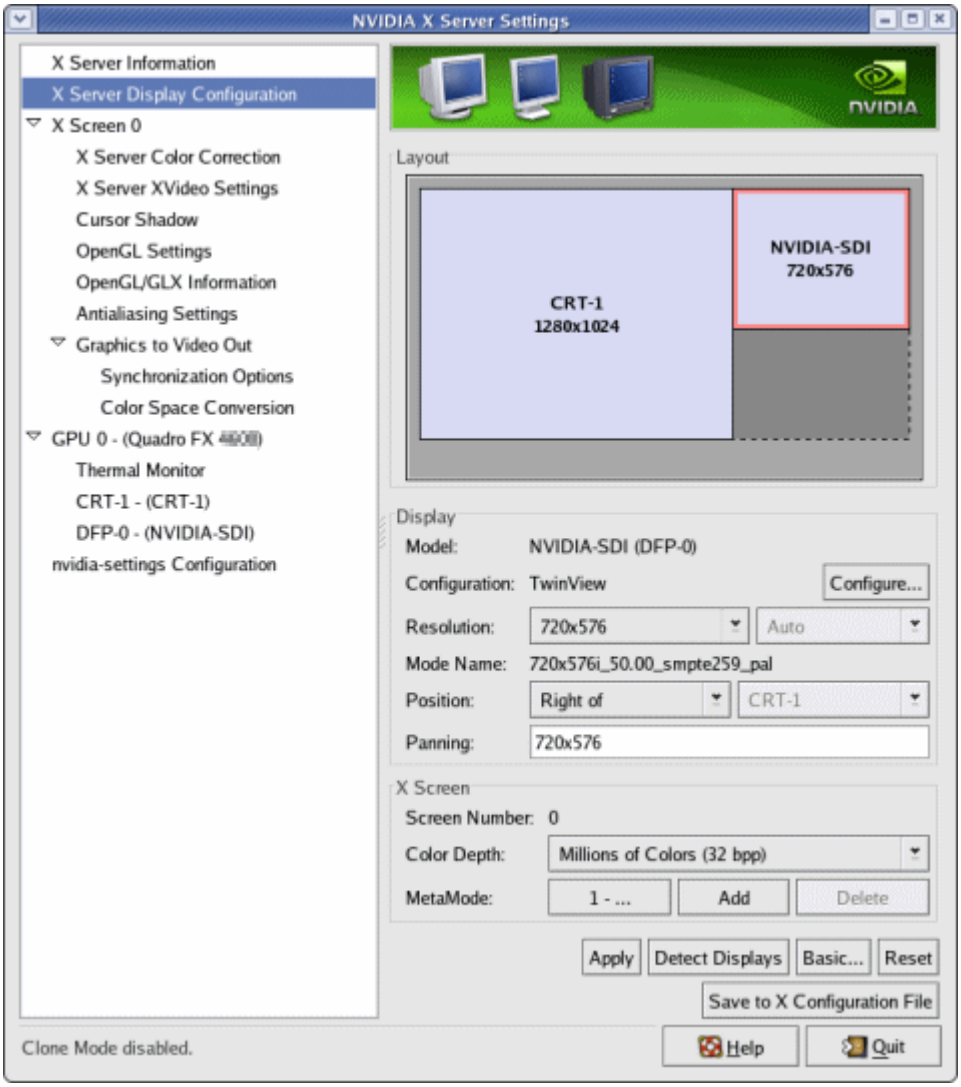
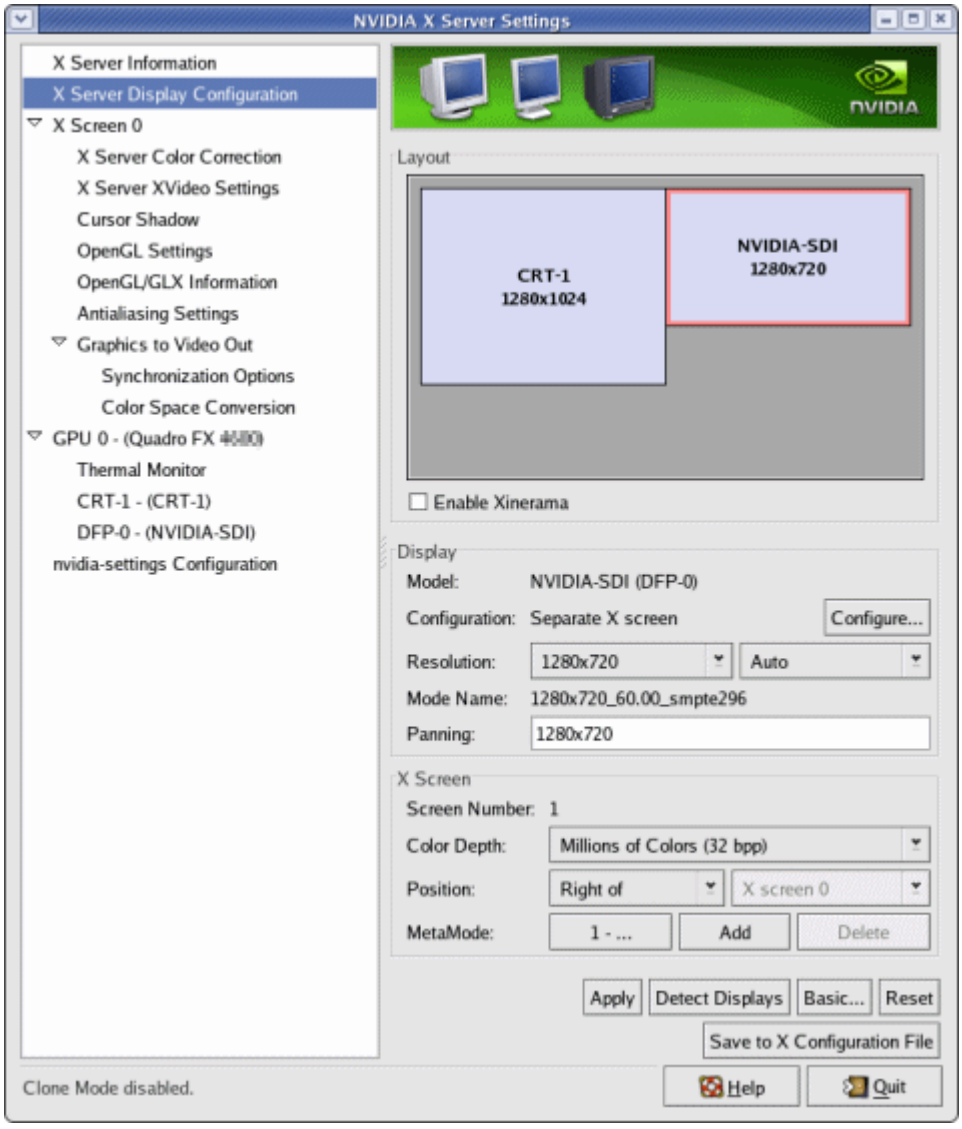


Figure 4.3 NVIDIA X Server Settings Page

2 Click **X Server Display Configuration** from the side view menu tree and then configure the display for Dualview mode.



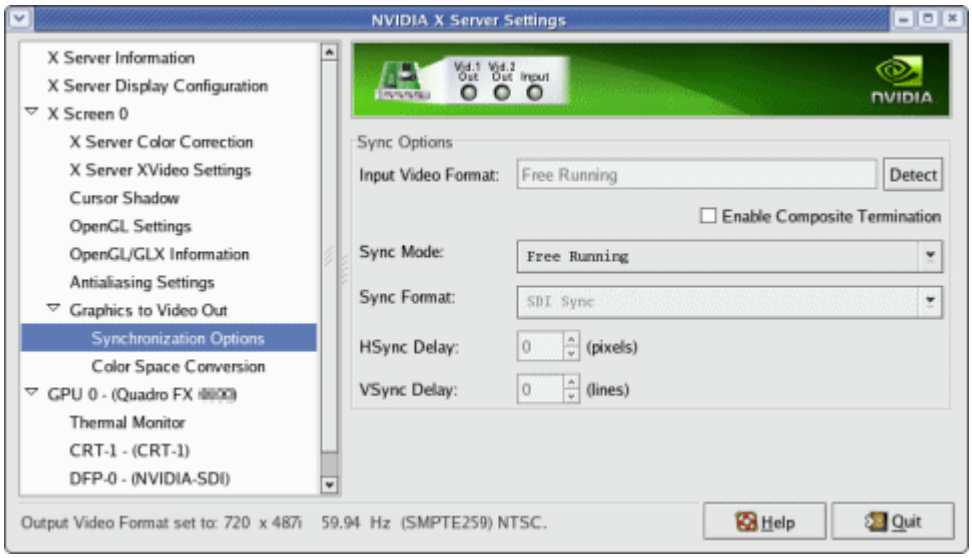
or x-screen mode



Step 2: Choose a Synchronization Method

1 Click the **Graphics to Video Out: Synchronization Options** tree item from the side menu.

The Sync Options page appears.



2 From the **Sync Options** group box, click the **Sync Mode** list arrow and then click the method you want to use to synchronize the SDI output:

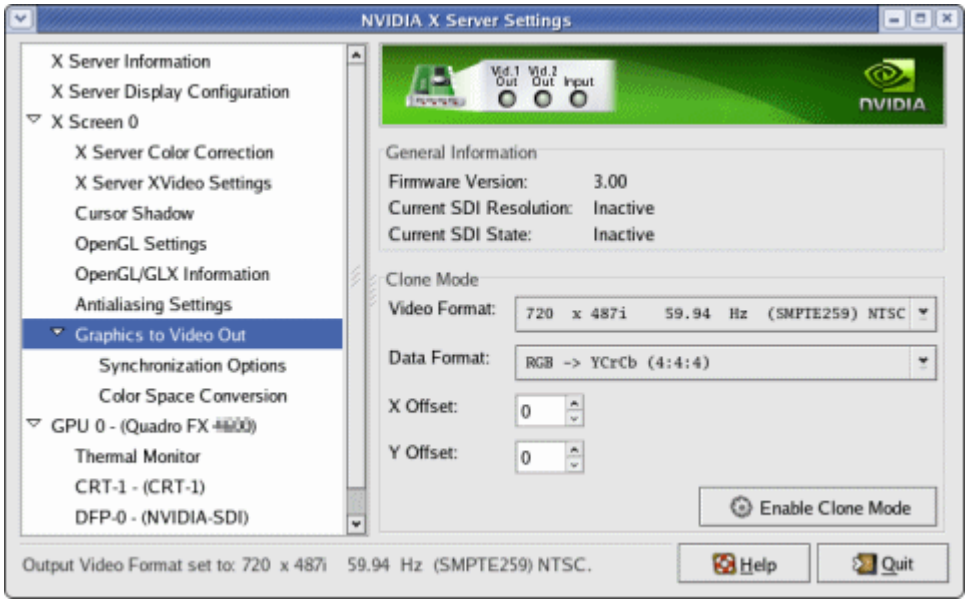
- **Free Running:** The SDI output will be synchronized with the timing chosen from the SDI signal format list.
- **Genlock:** The SDI output will be synchronized with the external sync signal.
- **Frame Lock:** The SDI output will be synchronized with the timing chosen from the SDI signal format list.

This list is limited to timings that can be synchronized with the detected external sync signal.

For more information regarding genlock and frame lock, see the section "Synchronizing the SDI Output to an External Source" on page 44.

Step 3: Choose Data Formats

1 Click the **Graphics to Video Out** tree item from the side menu.



2 Specify the Data Format

Click the **Output Data Format** arrow and then click the color format you want to use.

Data Format controls the color model, data packing, and alpha or z components in the outgoing video stream.

Note: The video format should already have been set up from the display configuration screen.

Advanced Setups

This section describes the following SDI controls and supplemental information:

Understanding the Status Indicators

The Graphics to SDI property page banner indicates the status of the SDI output as well as the external synchronization signals. Figure 4.4 shows the correlation between the indicators on the banner and the actual connectors.

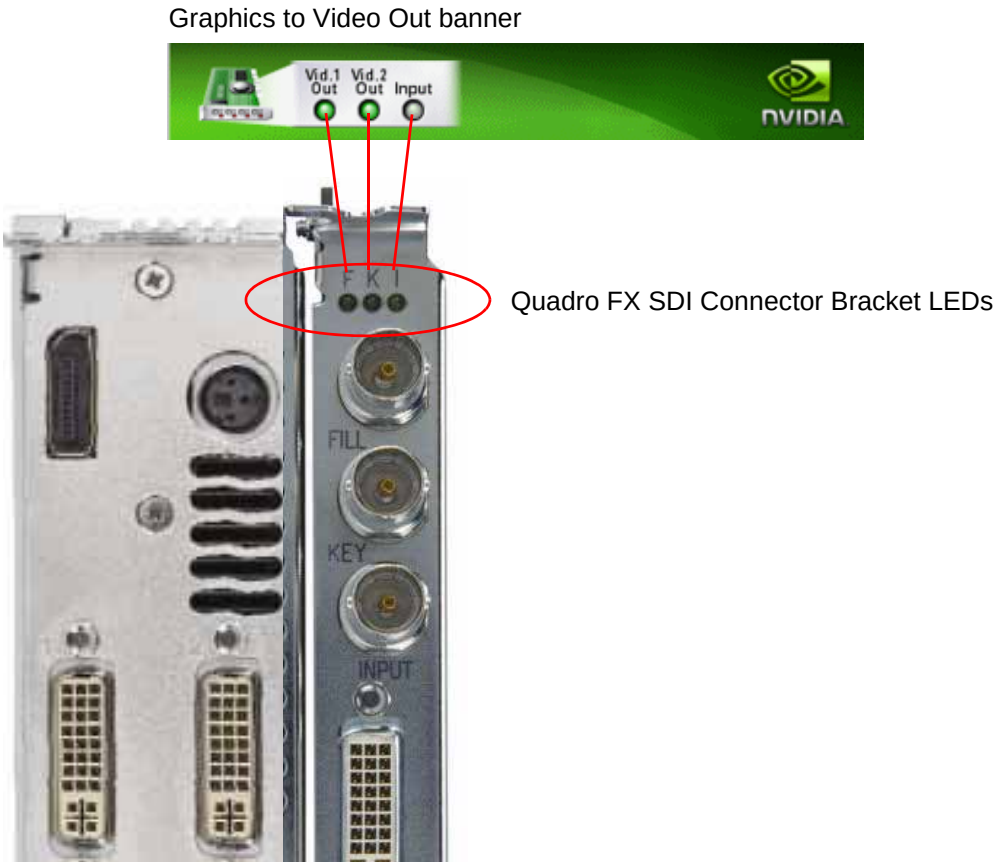


Figure 4.4 Connection Status Indicators

The activity of the LED graphics indicates the signal status as follows:

- **Vid. 1 Out** or **Vid. 2 Out**

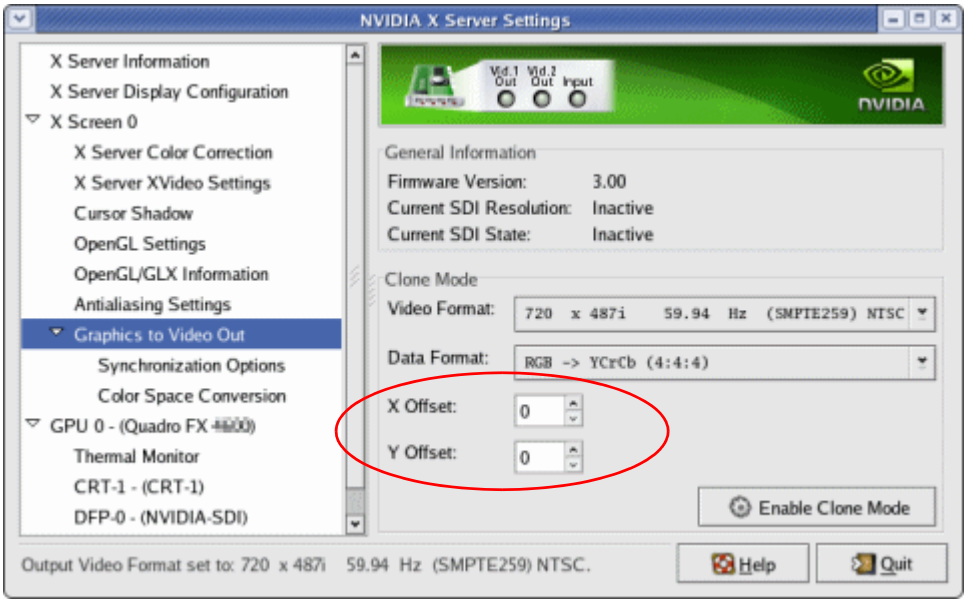
Status	Meaning
Off (gray)	SDI output is not in use
Steady Green	SDI output has power.
Blinking Green	SDI output is active.

- **Input**

Status	Meaning
Off (gray)	SDI input synchronization is disabled.
Blinking Green	Valid SDI synchronization signal is detected.

Adjusting the Desktop Area

By default, the entire desktop is converted to SDI output. If the desktop is smaller than the size of the SDI output, it will be scaled to fit. If the desktop is larger than the SDI output, it will be cropped to fit. Instead of using the entire desktop, you can specify a region of the desktop to convert to SDI output as follows:



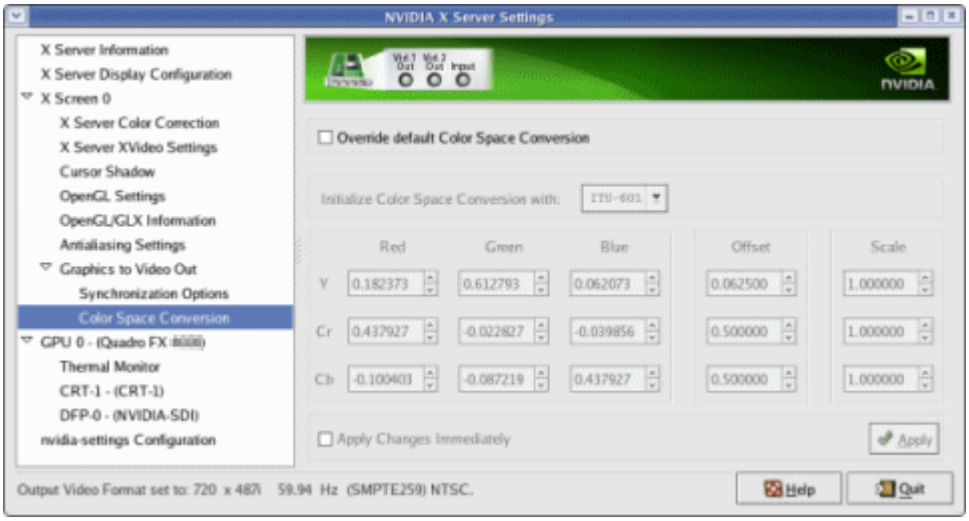
On the main Graphics to Video Out page, adjust the region size by specifying the **X Offset** and **Y Offset** values. The **X** and **Y** values indicate the pixel distance of the upper left corner of the output box from the upper left corner of the desktop.

Customizing the Color Space Conversion

To set your own RGB color space conversion:

1 Click the **Color Space Conversion** tree item from the side menu.

The Color Space Conversion page appears.



2 Check **Override default Color Space Conversion**.

3 Click the **Initialize Color Space Conversion with** list arrow and then click one of the standards to use as a starting point: ITU-601, 709, 177, or Identity.

4 Either enter values directly in the text boxes or use the corresponding up and down arrows to change any of the settings.

5 Click **Apply** to apply the settings.

To apply the settings as you change them, check **Apply Changes Immediately**.

Synchronizing the SDI Output to an External Source

You can synchronize the SDI output with other equipment in a broadcast or post production environment.

Genlock Versus Frame Lock

The **Graphics to SDI** page provides two methods for synchronizing the SDI output to a common sync source—Genlock and Frame lock.

Using Genlock

Genlock synchronizes the pixel scanning of the SDI output to an external synchronization source.

When using genlock, the SDI refresh rate is determined by the sync source, so any refresh rates that you may have chosen in the **Output Video Format** list do not apply.

Using Frame Lock

Frame lock synchronizes the frame rate of the SDI output to an external synchronization source.

When using frame lock, only modes that are valid for the frame rate of the sync source can be used for the SDI output. The valid modes will appear in the **Output Video Format** list.

Supported Synchronization Signals

NVIDIA Genlock supports the following external synchronization signal types:

- SDI
- Composite Bi-level (NTSC or PAL sources use bi-level composite signals.)
- Composite Tri-level (HDTV sources commonly use tri-level composite signals.)

Synchronization Instructions

Basic Setup

The following are the basic steps to synchronize the SDI output.

1 Connect the external sync source to the appropriate BNC connector on the graphics card.

See "Understanding the Connections" on page 10 for instructions on connecting the external sync signal to the graphics card.

2 Configure the sync source.

a Open the **Graphics to Video Out: Synchronization Options** page.

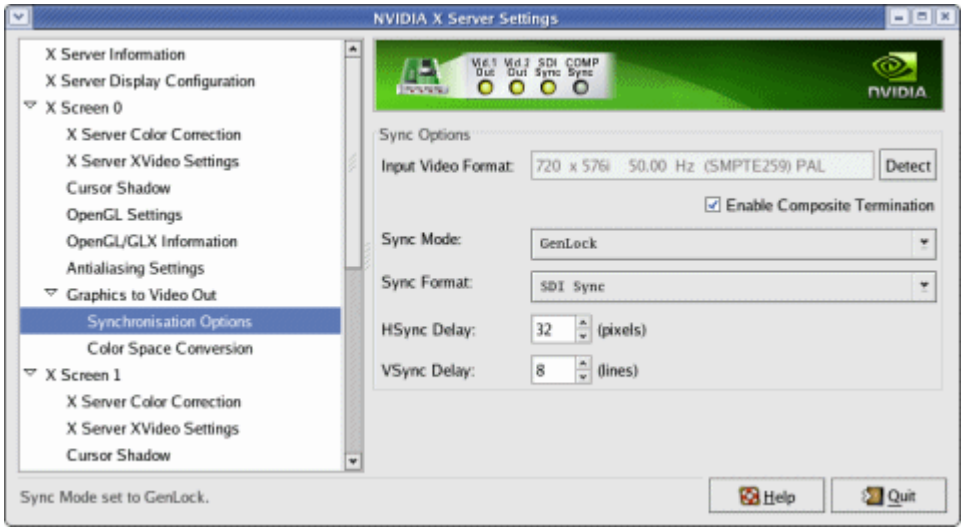


Figure 4.5 Synchronization Options Page

a Click the **Sync Mode** list arrow and then click either **Genlock** or **Framelock** synchronizing modes.

b Click the Sync Format list arrow and then click the format that matches external sync source that you connected - SDI Sync or Composite.

The software should automatically detect the external sync signal. When it does, the sync format information appears in the **Input Video Format** text box.

Detect If the software loses the external sync signal or does not detect it automatically, click **Detect** to force detection of the sync signal.

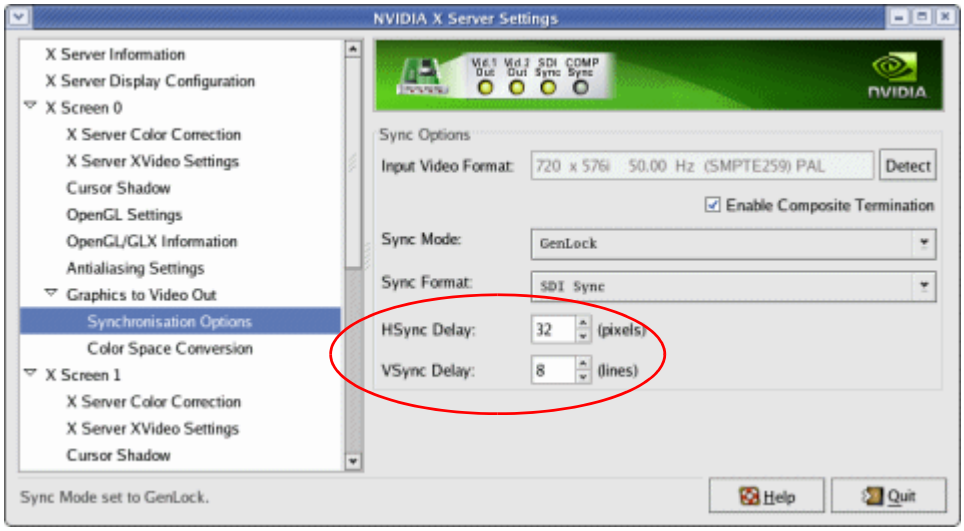
c If you chose frame lock synchronization, select the signal format you want to use as described under Step 3: Choose the Video and Data Formats.

Only those modes that are compatible with the detected sync signal will appear in the SDI signal format list.

Adding a Delay to the Signal

You can introduce a slight delay in the genlocked or frame locked SDI output. For example, if delivery of video from other equipment is delayed because of greater cable length, you can introduce a delay in the SDI output from this card so that both deliveries are in sync. To introduce a synchronization delay:

1 Open the **Graphics to Video Out** page and click **Synchronization Options**.



2 In the Synchronization Delay group box, introduce delays in the HSYNC and VSYNC signals as needed by clicking the appropriate up and down arrows.

You can also enter values directly into the text boxes.

CHAPTER

API CONTROL

The SDI application programming interface allows OpenGL or Direct3D applications to have full and exclusive control of the SDI output. This method of controlling the SDI output is also known as extended mode.

This chapter gives a brief introduction to this method of implementing graphics to SDI, and includes the following sections:

Refer to the following documents for additional information on using the APIs:

- *Programming the NVIDA Quadro FX 4800/5800 SDI*
- The *NVGVOSDK* , which can be obtained from NVIDIA.

SDI Application Programming Overview

Application programming of the NVIDIA Quadro FX SDI consists of two principle parts– device control and data transfer.

- **Device contro**l handles the hardware configuration as well as the starting and stopping of data transfers.

  This chapter covers the APIs related to data control.

- **Data transfer** is the sequence of operations that send graphics data to the video device for output.

Under WindowsXP

- **Device control** is handled by the NvGvo API, described in this chapter.
- **Data transfer** operations are performed by the OpenGL extension WGL_NV_video_out.

Under Linux

- **Device control** is handled by the NV-CONTROL X extension, described in this chapter.
- **Data transfer** operations are performed by the OpenGL extension GLX_NV_video_output.

Windows XP NvGvo API Description

This section describes the NvGvo APIs inthe following sections:

- "NvGvo Function Description" on page 49
- "NvGvo Structures, Enumerations, and Defines" on page 56

NvGvo Function Description

Table 5.1 NvGvo Function Index

Call	Description
`NvGvoCaps()`	Determine the graphics-to-video capabilities of the graphics card.
`NvGvoOpen()`	Open the graphics card for graphics-to-video operations using the OpenGL application interface.
`NvGvoClose()`	Close the graphics card for graphics-to-video operations using the OpenGL application interface.
`NvGvoDesktopOpen()`	Open the graphics cards for graphics-to-vVideo operations using the Desktop transparent mode interface.
`NvGvoDesktopClose()`	Close the graphics cards for graphics-to-video operations using the Desktop transparent mode interface.
`NvGvoStatus()`	Get the graphics-to-video status.
`NvGvoSyncFormatDetect()`	Detect the video format of the iincoming sync signal.
`NvGvoConfigGet()`	Get the current graphics-to-video configuration.
`NvGvoConfigSet()`	Set the graphics-to-video configuration.
`NvGvoIsRunning()`	Determine if there is an SDI out video stream.
`NvGvoStart()`	Start the SDI out video stream.
`NvGvoStop()`	Stop the SDI out video stream.
`NvGvoEnumSignalFormats()`	Enumerate the supported SDI signal formats.
`NvGvoIsFrameLockModeCompatible()`	Verify whether a mode is compatible with frame lock mode.
`NvGvoEnumDataFormats()`	Enumerate the supported SDI data formats.

NvGvoCaps()

```
//--------------------------------------------------------------------
// Function:    NvGvoCaps
// Description: Determine graphics adapter Graphics to Video capabilities.
// Parameters:  nAdapterNumber  - Graphics adapter number
//              nReserved       - Reserved (must be set to zero)
//              pAdapterCaps    - Pointer to receive capabilities
// Returns:     NV_OK           - Success
//              NV_NOTSUPPORTED - Graphics to Video not supported
//--------------------------------------------------------------------
NVRESULT NVAPIENTRY NvGvoCaps(UINT       nAdapterNumber IN,
                              UINT       nReserved      IN,
                              NVGVOCAPS* pAdapterCaps   OUT);
```

NvGvoOpen()

```
//--------------------------------------------------------------------
// Function:    NvGvoOpen
// Description: Open graphics adapter for Graphics to Video operations
//              using the OpenGL application interface.  Read operations
//              are permitted in this mode by multiple clients, but Write
//              operations are application exclusive.
// Parameters:  nAdapterNumber  - Graphics adapter number
//              nReserved       - Reserved (must be set to zero)
//              dwClass         - Class interface (NVGVOCLASS_* value)
//              dwAccessRights  - Access rights (NVGVO_O_* mask)
//              phGvoHandle     - Pointer to receive handle
// Returns:     NV_OK           - Success
//              NV_ACCESSDENIED - Access denied for requested access
//--------------------------------------------------------------------
NVRESULT NVAPIENTRY NvGvoOpen(UINT                 nAdapterNumber IN,
                              UINT                 nReserved      IN,
                              DWORD                dwClass        IN,
                              DWORD                dwAccessRights IN,
                              NVGVOHANDLE*         phGvoHandle    OUT);
```

NvGvoClose()

```
//--------------------------------------------------------------------
// Function:    NvGvoClose
// Description: Closes graphics adapter for Graphics to Video operations
```

```
//              using the OpenGL application interface.  Closing an
//              OpenGL handle releases the device.
// Parameters:  hGvoHandle - Handle to graphics adapter
// Returns:     NV_OK      - Success
//---------------------------------------------------------------------
NVRESULT NVAPIENTRY NvGvoClose(NVGVOHANDLE hGvoHandle IN);
```

NvGvoDesktopOpen()

```
//---------------------------------------------------------------------
// Function:    NvGvoDesktopOpen
// Description: Open graphics adapter for Graphics to Video operations
//              using the Desktop transparent mode interface. Read
//              operations are permitted in this mode by multiple clients,
//              but write operations are application exclusive.
// Parameters:  nAdapterNumber  - Graphics adapter number
//              nReserved       - Reserved (must be set to zero)
//              dwClass         - Class interface (NVGVOCLASS_* value)
//              dwAccessRights  - Access rights (NVGVO_O_* mask)
//              phGvoHandle     - Pointer to receive handle
// Returns:     NV_OK           - Success
//              NV_ACCESSDENIED - Access denied for requested access
//---------------------------------------------------------------------
NVRESULT NVAPIENTRY NvGvoDesktopOpen(UINT                nAdapterNumber IN,
                                     UINT                nReserved      IN,
                                     DWORD               dwClass        IN,
                                     DWORD               dwAccessRights IN,
                                     NVGVOHANDLE*        phGvoHandle    OUT);
```

NvGvoDesktopClose()

```
//---------------------------------------------------------------------
// Function:    NvGvoDesktopClose
// Description: Closes graphics adapter for Graphics to Video operations
//              using the Desktop transparent mode interface.
// Parameters:  hGvoHandle  - Handle to graphics adapter
//              bGvoRelease - TRUE to release device when handle closes
//                            FALSE to remain in desktop mode when handle
//                                  closes (other clients can open using
//                                  NvGvoDesktopOpen and release using
//                                  NvGvoDesktopClose)
// Returns:     NV_OK       - Success
```

```
//--------------------------------------------------------------------
NVRESULT NVAPIENTRY NvGvoDesktopClose(NVGVOHANDLE hGvoHandle IN,
                                      BOOL        bRelease   IN);
```

NvGvoStatus()

```
//--------------------------------------------------------------------
// Function:    NvGvoStatus
// Description: Get Graphics to Video status.
// Parameters:  hGvoHandle - Handle to graphics adapter
// Returns:     NV_OK      - Success
//--------------------------------------------------------------------
NVRESULT NVAPIENTRY NvGvoStatus(NVGVOHANDLE  hGvoHandle IN,
                                NVGVOSTATUS* pStatus    OUT);
```

NvGvoSyncFormatDetect()

```
//--------------------------------------------------------------------
// Function:    NvGvoSyncFormatDetect
// Description: Detects Graphics to Video incoming sync video format.
// Parameters:  hGvoHandle - Handle to graphics adapter
//              pdwWait    - Pointer to receive milliseconds to wait
//                           before NvGvoStatus will return detected
//                           syncFormat.
// Returns:     NV_OK - Success
//--------------------------------------------------------------------
NVRESULT NVAPIENTRY NvGvoSyncFormatDetect(NVGVOHANDLE hGvoHandle IN,
                                          DWORD*      pdwWait    OUT);
```

NvGvoConfigGet()

```
//--------------------------------------------------------------------
// Function:    NvGvoConfigGet
// Description: Get Graphics to Video configuration.
// Parameters:  hGvoHandle - Handle to graphics adapter
//              pConfig    - Pointer to Graphics to Video configuration
// Returns:     NV_OK      - Success
//--------------------------------------------------------------------
NVRESULT NVAPIENTRY NvGvoConfigGet(NVGVOHANDLE  hGvoHandle IN,
                                   NVGVOCONFIG* pConfig    OUT);
```

NvGvoConfigSet()

```
//--------------------------------------------------------------------
// Function:    NvGvoConfigSet
// Description: Set Graphics to Video configuration.
// Parameters:  hGvoHandle      - Handle to graphics adapter
//              pConfig         - Pointer to Graphics to Video config
// Returns:     NV_OK           - Success
//              NV_ACCESSDENIED - Access denied (no write access)
//              NV_RUNNING      - Requested settings require NvGvoStop
//--------------------------------------------------------------------
NVRESULT NVAPIENTRY NvGvoConfigSet(NVGVOHANDLE        hGvoHandle IN,
                                   const NVGVOCONFIG* pConfig    IN);
```

NvGvoIsRunning()

```
//--------------------------------------------------------------------
// Function:    NvGvoIsRunning
// Description: Determine if Graphics to Video output is running.
// Parameters:  hGvoHandle      - Handle to graphics adapter
// Returns:     NV_RUNNING      - Graphics-to-Video is running
//              NV_NOTRUNNING   - Graphics-to-Video is not running
//--------------------------------------------------------------------
NVRESULT NVAPIENTRY NvGvoIsRunning(NVGVOHANDLE hGvoHandle IN);
```

NvGvoStart()

```
//--------------------------------------------------------------------
// Function:    NvGvoStart
// Description: Start Graphics to Video output.
// Parameters:  hGvoHandle      - Handle to graphics adapter
// Returns:     NV_OK           - Success
//              NV_ACCESSDENIED - Access denied (no write access)
//              NV_RUNNING      - Graphics to Video already running
//--------------------------------------------------------------------
NVRESULT NVAPIENTRY NvGvoStart(NVGVOHANDLE hGvoHandle IN);
```

NvGvoStop()

```
//--------------------------------------------------------------------
// Function:    NvGvoStop
// Description: Stop Graphics to Video output.
// Parameters:  hGvoHandle      - Handle to graphics adapter
// Returns:     NV_OK           - Success
//              NV_ACCESSDENIED - Access denied (no write access)
//              NV_NOTRUNNING   - Graphics to Video not running
//--------------------------------------------------------------------
NVRESULT NVAPIENTRY NvGvoStop(NVGVOHANDLE hGvoHandle IN);
```

NvGvoEnumSignalFormats()

```
//--------------------------------------------------------------------
// Function:    NvGvoEnumSignalFormats
// Description: Enumerate signal formats supported by Graphics to Video.
// Parameters:  hGvoHandle          - Handle to graphics adapter
//              nEnumIndex          - Enumeration index
//              bByEnum             - TRUE nEnumIndex is NVSIGNALFORMAT_*
//                                    FALSE nEnumIndex is 0..n-1
//              pSignalFormatDetail - Pointer to receive detail or NULL
// Returns:     NV_OK               - Success
//              NV_NOMORE           - No more signal formats to enumerate
//              NV_NOTSUPPORTED     - Unsupported NVSIGNALFORMAT_ enumeration
//--------------------------------------------------------------------
NVRESULT NVAPIENTRY NvGvoEnumSignalFormats(
         NVGVOHANDLE              hGvoHandle          IN,
         int                      nEnumIndex          IN,
         BOOL                     bByEnum             IN,
         NVGVOSIGNALFORMATDETAIL* pSignalFormatDetail OUT);
```

NvGvoIsFrameLockModeCompatible()

```
//--------------------------------------------------------------------
// Function:    NvGvoIsFrameLockModeCompatible
// Description: Checks whether modes are compatible in framelock mode
// Parameters:  hGvoHandle          - Handle to graphics adapter
//              nSrcEnumIndex       - Source Enumeration index
//              nDestEnumIndex      - Destination Enumeration index
//              pbCompatible        - Pointer to receive compatability
// Returns:     NV_OK               - Success
//              NV_NOTSUPPORTED     - Unsupported NVSIGNALFORMAT_ enumeration
//--------------------------------------------------------------------
NVRESULT NVAPIENTRY NvGvoIsFrameLockModeCompatible(
         NVGVOHANDLE              hGvoHandle          IN,
         int                      nSrcEnumIndex       IN,
         int                      nDestEnumIndex      IN,
         BOOL*                    pbCompatible        OUT);
```

NvGvoEnumDataFormats()

```
//--------------------------------------------------------------------
// Function:    NvGvoEnumDataFormats
// Description: Enumerate data formats supported by Graphics to Video.
// Parameters:  hGvoHandle        - Handle to graphics adapter
//              nEnumIndex        - Enumeration index
//              bByEnum           - TRUE nEnumIndex is NVDATAFORMAT_*
//                                  FALSE nEnumIndex is 0..n-1
//              pDataFormatDetail - Pointer to receive detail or NULL
// Returns:     NV_OK             - Success
//              NV_NOMORE         - No more data formats to enumerate
//              NV_NOTSUPPORTED   - Unsupported NVDATAFORMAT_ enumeration
//--------------------------------------------------------------------
NVRESULT NVAPIENTRY NvGvoEnumDataFormats(
         NVGVOHANDLE            hGvoHandle        IN,
         int                    nEnumIndex        IN,
         BOOL                   bByEnum           IN,
         NVGVODATAFORMATDETAIL* pDataFormatDetail OUT);
```

NvGvo Structures, Enumerations, and Defines

Miscellaneous Defines

```
typedef UINT NVGVOHANDLE;        // Handle from NvGvoOpen() or NvGvoDesktopOpen()
#define INVALID_NVGVOHANDLE  0   // Invalid NVGVOHANDLE

typedef DWORD NVGVOOWNERID;      // Unique identifier for owner of Graphics to
                                 // Video output (process identifier or
                                 // NVGVOOWNERID_NONE)
#define NVGVOOWNERID_NONE    0   // Unregistered ownerId

enum NVGVOOWNERTYPE              // Owner type for device
{
    NVGVOOWNERTYPE_NONE      ,   //  No owner for device
    NVGVOOWNERTYPE_OPENGL    ,   //  OpenGL application owns device
    NVGVOOWNERTYPE_DESKTOP   ,   //  Desktop transparent mode owns device
};

// Access rights for NvGvoOpen() or NvGvoDesktopOpen()
#define NVGVO_O_READ                0x00000000      // Read access
#define NVGVO_O_WRITE_EXCLUSIVE     0x00010001      // Write exclusive access
```

Video Signal Format and Resolution Enumerations

```
enum NVGVOSIGNALFORMAT
{
    NVGVOSIGNALFORMAT_ERROR = -1               , // Invalid signal format
    NVGVOSIGNALFORMAT_487I_5994_SMPTE259_NTSC  , // 01  487i  59.94Hz  (SMPTE259)
                                                 //                     NTSC
    NVGVOSIGNALFORMAT_576I_5000_SMPTE259_PAL   , // 02  576i  50.00Hz  (SMPTE259)
                                                   //                     PAL
    NVGVOSIGNALFORMAT_720P_5994_SMPTE296       , // 03  720p  59.94Hz  (SMPTE296)
    NVGVOSIGNALFORMAT_720P_6000_SMPTE296       , // 04  720p  60.00Hz  (SMPTE296)
    NVGVOSIGNALFORMAT_1035I_5994_SMPTE260      , // 05  1035i 59.94Hz  (SMPTE260)
    NVGVOSIGNALFORMAT_1035I_6000_SMPTE260      , // 06  1035i 60.00Hz  (SMPTE260)
    NVGVOSIGNALFORMAT_1080I_5000_SMPTE274      , // 08  1080i 50.00Hz  (SMPTE274)
    NVGVOSIGNALFORMAT_1080I_5994_SMPTE274      , // 09  1080i 59.94Hz  (SMPTE274)
    NVGVOSIGNALFORMAT_1080I_6000_SMPTE274      , // 10  1080i 60.00Hz  (SMPTE274)
```

```
    NVGVOSIGNALFORMAT_1080PSF_23976_SMPTE274    , // 11 1080PsF 23.976Hz
(SMPTE274)
    NVGVOSIGNALFORMAT_1080PSF_2400_SMPTE274     , // 12 1080PsF 24.00Hz
(SMPTE274)
    NVGVOSIGNALFORMAT_1080PSF_2500_SMPTE274     , // 13 1080PsF 25.00Hz
(SMPTE274)
    NVGVOSIGNALFORMAT_1080PSF_3000_SMPTE274     , // 14 1080PsF 30.00Hz
(SMPTE274)
    NVGVOSIGNALFORMAT_1080P_23976_SMPTE274      , // 15 1080p 23.976Hz (SMPTE274)
    NVGVOSIGNALFORMAT_1080P_2400_SMPTE274       , // 16 1080p 24.00Hz  (SMPTE274)
    NVGVOSIGNALFORMAT_1080P_2500_SMPTE274       , // 17 1080p 25.00Hz  (SMPTE274)
    NVGVOSIGNALFORMAT_1080P_2997_SMPTE274       , // 18 1080p 29.97Hz  (SMPTE274)
    NVGVOSIGNALFORMAT_1080P_3000_SMPTE274       , // 19 1080p 30.00Hz  (SMPTE274)
    NVGVOSIGNALFORMAT_1080PSF_2997_SMPTE274     , // 20 1080PsF 29.97Hz
(SMPTE274)

    NVGVOSIGNALFORMAT_720P_5000_SMPTE296        , // 21 720p  50.00Hz  (SMPTE296)
    NVGVOSIGNALFORMAT_720P_3000_SMPTE296        , // 22 720p  30.00Hz  (SMPTE296)
    NVGVOSIGNALFORMAT_720P_2997_SMPTE296        , // 23 720p  29.97Hz  (SMPTE296)
    NVGVOSIGNALFORMAT_720P_2500_SMPTE296        , // 24 720p  25.00Hz  (SMPTE296)
    NVGVOSIGNALFORMAT_720P_2400_SMPTE296        , // 25 720p  24.00Hz  (SMPTE296)
    NVGVOSIGNALFORMAT_720P_2398_SMPTE296        , // 26 720p  23.98Hz  (SMPTE296)

    NVGVOSIGNALFORMAT_1080I_4800_SMPTE274       , // 27 1080i 48.00Hz  (SMPTE296)
    NVGVOSIGNALFORMAT_1080I_4796_SMPTE274       , // 28 1080i 47.96Hz  (SMPTE296)
    NVGVOSIGNALFORMAT_1080PSF_2398_SMPTE274     , // 29 1080PsF 23.98Hz
(SMPTE296)

    NVGVOSIGNALFORMAT_2048P_3000_SMPTE372       , // 30 2048P 30.00Hz  (SMPTE372)
    NVGVOSIGNALFORMAT_2048P_2997_SMPTE372       , // 31 2048P 29.97Hz  (SMPTE372)
    NVGVOSIGNALFORMAT_2048I_6000_SMPTE372       , // 32 2048I 60.00Hz  (SMPTE372)
    NVGVOSIGNALFORMAT_2048I_5994_SMPTE372       , // 33 2048I 59.94Hz  (SMPTE372)
    NVGVOSIGNALFORMAT_2048P_2500_SMPTE372       , // 34 2048P 25.00Hz  (SMPTE372)
    NVGVOSIGNALFORMAT_2048I_5000_SMPTE372       , // 35 2048I 50.00Hz  (SMPTE372)
    NVGVOSIGNALFORMAT_2048P_2400_SMPTE372       , // 36 2048P 24.00Hz  (SMPTE372)
    NVGVOSIGNALFORMAT_2048I_4800_SMPTE372       , // 37 2048I 48.00Hz  (SMPTE372)
    NVGVOSIGNALFORMAT_2048P_2398_SMPTE372       , // 38 2048P 23.98Hz  (SMPTE372)
    NVGVOSIGNALFORMAT_2048I_4796_SMPTE372       , // 39 2048I 23.98Hz  (SMPTE372)

    NVGVOSIGNALFORMAT_END                        // 40 To indicate end of signal
format list
};
```

SMPTE Standards Format Enumeration

```
enum NVVIDEOSTANDARD
{
    NVVIDEOSTANDARD_SMPTE259                 ,  // SMPTE259
    NVVIDEOSTANDARD_SMPTE260                 ,  // SMPTE260
    NVVIDEOSTANDARD_SMPTE274                 ,  // SMPTE274
    NVVIDEOSTANDARD_SMPTE295                 ,  // SMPTE295
    NVVIDEOSTANDARD_SMPTE296                 ,  // SMPTE296
    NVVIDEOSTANDARD_SMPTE372                 ,  // SMPTE372
};
```

HD or SD Video Type Enumeration

```
enum NVVIDEOTYPE
{
    NVVIDEOTYPE_SD                          ,   // Standard-definition (SD)
    NVVIDEOTYPE_HD                          ,   // High-definition     (HD)
};
```

Interlace Mode Enumeration

```
enum NVINTERLACEMODE
{
    NVINTERLACEMODE_PROGRESSIVE            ,   // Progressive              (p)
    NVINTERLACEMODE_INTERLACE              ,   // Interlace                (i)
    NVINTERLACEMODE_PSF                    ,   // Progressive Segment Frame (psf)
};
```

Video Data Format Enumeration

```
enum NVGVODATAFORMAT
{
    NVGVODATAFORMAT_UNKNOWN = -1             ,
    NVGVODATAFORMAT_R8G8B8_TO_YCRCB444       , // R8:G8:B8  => YCrCb  (4:4:4)
    NVGVODATAFORMAT_R8G8B8A8_TO_YCRCBA4444   , // R8:G8:B8:A8 => YCrCbA (4:4:4:4)
    NVGVODATAFORMAT_R8G8B8Z10_TO_YCRCBZ4444  , // R8:G8:B8:Z10
                                               // => YCrCbZ (4:4:4:4)
    NVGVODATAFORMAT_R8G8B8_TO_YCRCB422       , // R8:G8:B8  => YCrCb  (4:2:2)
    NVGVODATAFORMAT_R8G8B8A8_TO_YCRCBA4224   , // R8:G8:B8:A8 => YCrCbA (4:2:2:4)
    NVGVODATAFORMAT_R8G8B8Z10_TO_YCRCBZ4224  , // R8:G8:B8:Z10
                                               // => YCrCbZ  4:2:2:4)
```

```
    NVGVODATAFORMAT_R8G8B8_TO_RGB444          , // R8:G8:B8  => RGB   (4:4:4)
    NVGVODATAFORMAT_R8G8B8A8_TO_RGBA4444      , // R8:G8:B8:A8 => RGBA (4:4:4:4)
    NVGVODATAFORMAT_R8G8B8Z10_TO_RGBZ4444     , // R8:G8:B8:Z10 => RGBZ (4:4:4:4)
    NVGVODATAFORMAT_Y10CR10CB10_TO_YCRCB444   , // Y10:CR10:CB10
                                                //=> YCrCb  (4:4:4)
    NVGVODATAFORMAT_Y10CR8CB8_TO_YCRCB444       , // Y10:CR8:CB8 => YCrCb (4:4:4)
    NVGVODATAFORMAT_Y10CR8CB8A10_TO_YCRCBA4444  , // Y10:CR8:CB8:A10
                                                  // => YCrCbA  (4:4:4:4)
    NVGVODATAFORMAT_Y10CR8CB8Z10_TO_YCRCBZ4444  , // Y10:CR8:CB8:Z10
                                                  // => YCrCbZ  (4:4:4:4)
    NVGVODATAFORMAT_DUAL_R8G8B8_TO_DUAL_YCRCB422    , // R8:G8:B8 + R8:G8:B8
                                                      // => YCrCb (4:2:2 + 4:2:2)
    NVGVODATAFORMAT_DUAL_Y8CR8CB8_TO_DUAL_YCRCB422  , // Y8:CR8:CB8 + Y8:CR8:CB8
                                                      // => YCrCb (4:2:2 + 4:2:2)
    NVGVODATAFORMAT_R10G10B10_TO_YCRCB422       , // R10:G10:B10 => YCrCb (4:2:2)
    NVGVODATAFORMAT_R10G10B10_TO_YCRCB444       , // R10:G10:B10 => YCrCb (4:4:4)
    NVGVODATAFORMAT_Y12CR12CB12_TO_YCRCB444     , // Y12:CR12:CB12
                                                  // => YCrCb (4:4:4)
    NVGVODATAFORMAT_Y12CR12CB12_TO_YCRCB422     , // Y12:CR12:CB12
                                                  // => YCrCb  (4:2:2)
    NVGVODATAFORMAT_Y10CR10CB10_TO_YCRCB422     , // Y10:CR10:CB10
                                                  // => YCrCb  (4:2:2)
    NVGVODATAFORMAT_Y8CR8CB8_TO_YCRCB422        , // Y8:CR8:CB8
                                                  // => YCrCb  (4:2:2)
    NVGVODATAFORMAT_Y10CR8CB8A10_TO_YCRCBA4224  , // Y10:CR8:CB8:A10
                                                  // => YCrCbA (4:2:2:4)
    NVGVODATAFORMAT_R10G10B10_TO_RGB444         ,  // R10:G10:B10 => RGB (4:4:4)
    NVGVODATAFORMAT_R12G12B12_TO_RGB444         ,  // R12:G12:B12 => RGB (4:4:4)
};
```

Video Output Area Enumeration

```
enum NVGVOOUTPUTAREA
{
    NVGVOOUTPUTAREA_FULLSIZE      , // Output to entire video resolution
                                      (full size)
    NVGVOOUTPUTAREA_SAFEACTION    , // Output to centered 90% of video resolution
                                      (safe action)
    NVGVOOUTPUTAREA_SAFETITLE     , // Output to centered 80% of video resolution
                                      (safe title)
};
```

Synchronization Source Enumeration

```
enum NVGVOSYNCSOURCE
{
    NVGVOSYNCSOURCE_SDISYNC        ,   // SDI Sync  (Digital input)
    NVGVOSYNCSOURCE_COMPSYNC       ,   // COMP Sync (Composite input)
};
```

Composite Synchronization Type Enumeration

```
enum NVGVOCOMPSYNCTYPE
{
    NVGVOCOMPSYNCTYPE_AUTO         ,   // Auto-detect
    NVGVOCOMPSYNCTYPE_BILEVEL      ,   // Bi-level signal
    NVGVOCOMPSYNCTYPE_TRILEVEL     ,   // Tri-level signal
};
```

Video Output Status Enumeration

```
enum NVGVOOUTPUTSTATUS
{
    NVGVOOUTPUTSTATUS_OFF       ,  // Output not in use
    NVGVOOUTPUTSTATUS_ERROR     ,  // Error detected
    NVGVOOUTPUTSTATUS_SDI_SD    ,  // SDI output (standard-definition)
    NVGVOOUTPUTSTATUS_SDI_HD    ,  // SDI output (high-definition)
};
```

Synchronization Input Status Enumeration

```
enum NVGVOSYNCSTATUS
{
    NVGVOSYNCSTATUS_OFF         ,  // Sync not detected
    NVGVOSYNCSTATUS_ERROR       ,  // Error detected
    NVGVOSYNCSTATUS_SYNCLOSS    ,  // Genlock in use, format mismatch with output
    NVGVOSYNCSTATUS_COMPOSITE   ,  // Composite sync
    NVGVOSYNCSTATUS_SDI_SD      ,  // SDI sync (standard-definition)
    NVGVOSYNCSTATUS_SDI_HD      ,  // SDI sync (high-definition)
};
```

Device Capabilities Defines

```
#define NVGVOCAPS_VIDOUT_SDI     0x00000001  // Supports Serial Digital Interface
```

```
                                                (SDI) output
#define NVGVOCAPS_SYNC_INTERNAL  0x00000100  // Supports Internal timing source
#define NVGVOCAPS_SYNC_GENLOCK   0x00000200  // Supports Genlock timing source
#define NVGVOCAPS_SYNCSRC_SDI    0x00001000  // Supports Serial Digital Interface
                                                (SDI) synchronization input
#define NVGVOCAPS_SYNCSRC_COMP   0x00002000  // Supports Composite
                                                synchronization input
#define NVGVOCAPS_OUTPUTMODE_DESKTOP   0x00010000   // Supports Desktop
                                                       transparent mode
#define NVGVOCAPS_OUTPUTMODE_OPENGL    0x00020000   // Supports OpenGL
                                                       application mode
#define NVGVOCLASS_SDI           0x00000001  // SDI-class interface:
                                               SDI output with two genlock inputs
```

Driver Version Structure

```
struct  NVGVODRIVER
{                                       // Driver version
    WORD    wMajorVersion;              //  Major version
    WORD    wMinorVersion;              //  Minor version
    WORD    wRevision;                  //  Revision
    WORD    wBuild;                     //  Build
};
```

Firmware Version Structure

```
struct  NVGVOFIRMWARE
{                                       // Firmware version
    WORD          wMajorVersion;        //  Major version
    WORD          wMinorVersion;        //  Minor version
};
```

Device Capabilities Structure

```
struct  NVGVOCAPS
{
    WORD              cbSize;     // Caller sets to sizeof(NVGVOCAPS)
    char              szAdapterName[NVADAPTERNAME_MAXLEN];
                                  // Graphics adapter name
    DWORD             dwClass;    // Graphics adapter classes
                                  // (NGVOCLASS_* mask)
    DWORD             dwCaps;     // Graphics adapter capabilities
```

```
                                    // (NVGVOCAPS_* mask)
    DWORD             dwDipSwitch; // On-board DIP switch settings bits
    DWORD             dwDipSwitchReserved;
                                    // On-board DIP switch settings reserved bits

    NVGVODRIVER       Driver;   // Driver version
                                // (see Driver Version Structure)
    NVGVOFIRMWARE     Firmware; // Firmware version
                                // (see Firmware Version Structure)

    NVGVOOWNERID      ownerId;    // Unique identifier for owner of video output
                                  // (NVGVOOWNERID_NONE if free running)
    NVGVOOWNERTYPE    ownerType;  // Owner type for video output
                                  // (OpenGL application or Desktop mode)
};
```

Device Status Structure

```
struct  NVGVOSTATUS
{
    WORD              cbSize;     // Caller sets to sizeof(NVGVOSTATUS)
    NVGVOOUTPUTSTATUS vid1Out;    // Video 1 output status
    NVGVOOUTPUTSTATUS vid2Out;    // Video 2 output status
    NVGVOSYNCSTATUS   sdiSyncIn;  // SDI sync input status
    NVGVOSYNCSTATUS   compSyncIn; // Composite sync input status
    BOOL              syncEnable; // Sync enable (TRUE if using syncSource)
    NVGVOSYNCSOURCE   syncSource; // Sync source
    NVGVOSIGNALFORMAT syncFormat; // Sync format
    NVGVOOWNERID      ownerId;    // Unique identifier for owner of video output
    NVGVOOWNERTYPE    ownerType;  // Owner type for video output
                                  // (OpenGL application or Desktop mode)
    BOOL   bframeLockEnable;         // Framelock enable flag
    BOOL   bOutputVideoLocked;       // Output video timing locked status
    int    nDataIntegrityCheckErrorCount; // Data integrity check error count
    BOOL   bDataIntegrityCheckEnabled;    // Data integrity check status enabled
    BOOL   bDataIntegrityCheckFailed;     // Data integrity check status failed
    BOOL   bSyncSourceLocked;        // genlocked to framelocked to ref signal
    BOOL   bPowerOn;                 // TRUE: indicates there is sufficient power
};
```

Output Region Structure

```
struct NVGVOOUTPUTREGION
{
    WORD              x;                // Horizontal origin in pixels
    WORD              y;                // Vertical origin in pixels
    WORD              width;            // Width of region in pixels
    WORD              height;           // Height of region in pixels
};
```

Gamma Ramp (8-bit Index) Structure

```
typedef struct NVGAMMARAMP8
{
    WORD              cbSize;         // Caller sets to sizeof(NVGAMMARAMP8)
    WORD              wRed[256];      // Red channel gamma ramp
                                         (8-bit index, 16-bit values)
    WORD              wGreen[256];    // Green channel gamma ramp
                                         (8-bit index, 16-bit values)
    WORD              wBlue[256];     // Blue channel gamma ramp
                                         (8-bit index, 16-bit values)
} NVGAMMARAMP8;
```

Gamma Ramp (10-bit Index) Structure

```
typedef struct NVGAMMARAMP10
{
    WORD              cbSize;         // Caller sets to sizeof(NVGAMMARAMP10)
    WORD              wRed[1024];     // Red channel gamma ramp
                                         (10-bit index, 16-bit values)
    WORD              wGreen[1024];   // Green channel gamma ramp
                                         (10-bit index, 16-bit values)
    WORD              wBlue[1024];    // Blue channel gamma ramp
                                         (10-bit index, 16-bit values)
} NVGAMMARAMP10;
```

Sync Delay Structure

```
typedef struct tagNVGVOSYNCDELAY
{
    WORD              wHorizontalDelay;     // Horizontal delay in pixels
    WORD              wVerticalDelay;       // Vertical delay in lines
```

```
} NVGVOSYNCDELAY;
```

Video Mode Information Structure

```
typedef struct NVVIDEOMODE
{
    DWORD            dwHorizontalPixels;   // Horizontal resolution (in pixels)
    DWORD            dwVerticalLines; // Vertical resolution for frame (in lines)
    NVFLOAT          fFrameRate;           // Frame rate
    NVINTERLACEMODE  interlaceMode;        // Interlace mode
    NVVIDEOSTANDARD  videoStandard;        // SMPTE standards format
    NVVIDEOTYPE      videoType;            // HD or SD signal classification
};
```

Signal Format Details Structure

```
struct NVGVOSIGNALFORMATDETAIL
{
    WORD              cbSize;             // Caller sets to
                                             sizeof(NVGVOSIGNALFORMATDETAIL)
    NVGVOSIGNALFORMAT signalFormat;       // Signal format enumerated value
    char              szValueName[NVVALUENAME_MAXLEN];
                      // Signal format name, in the form:
                      //  <name>\t<rate>\tHz\t(<standard>)[\t<description>]
                      //  "480i\t59.94\tHz\t(SMPTE259)\tNTSC"
                      //  "1080i\t50.00\tHz\t(SMPTE274)"
    char              szAlternateName[NVVALUENAME_MAXLEN];
                      // Signal format alternate name (or empty string):
                      //  "1080PsF\t25.00\tHz\t(SMPTE274)"
    NVVIDEOMODE       videoMode;          // Video mode for signal format
};
```

P-Buffer Format Defines

```
#define NVGVOPBUFFERFORMAT_R8G8B8                0x00000001   // R8:G8:B8
#define NVGVOPBUFFERFORMAT_R8G8B8Z24             0x00000002   // R8:G8:B8:Z24
#define NVGVOPBUFFERFORMAT_R8G8B8A8              0x00000004   // R8:G8:B8:A8
#define NVGVOPBUFFERFORMAT_R8G8B8A8Z24           0x00000008  // R8:G8:B8:A8:Z24
#define NVGVOPBUFFERFORMAT_R16FPG16FPB16FP       0x00000010  // R16FP:G16FP:B16FP
#define NVGVOPBUFFERFORMAT_R16FPG16FPB16FPZ24    0x00000020
```

```
                                                    //  R16FP:G16FP:B16FP:Z24
#define NVGVOPBUFFERFORMAT_R16FPG16FPB16FPA16FP     0x00000040
                                                    // R16FP:G16FP:B16FP:A16FP
#define NVGVOPBUFFERFORMAT_R16FPG16FPB16FPA16FPZ24 0x00000080
                                                    // R16FP:G16FP:B16FP:A16FP:Z24
```

Data Format Details Structure

```
struct NVGVODATAFORMATDETAIL
{
    WORD              cbSize;             // Caller sets to
                                             sizeof(NVGVODATAFORMATDETAIL)
    NVGVODATAFORMAT   dataFormat;         // Data format enumerated value
    DWORD             dwCaps;             // Data format capabilities
                                             (NVGVOCAPS_* mask)
    struct
    {
        DWORD         dwPbufferFormats;   // Supported p-buffer formats
                                             (NVGVOPBUFFERFORMAT_* mask)
        DWORD         dwPbufferCount;     // Number of p-buffers
        char          szValueName[NVVALUENAME_MAXLEN];
                                          // Data format input name, in the form:
                                          //  <name>
                                          //  "R8:G8:B8:A8"
    } in;
    struct
    {
        char          szValueName[NVVALUENAME_MAXLEN];
                                        // Data format output name, in the form:
                                        //  <name>\t<format>
                                        //  "YCrCbA\t(4:2:2:4)"
    } out;
};
```

Device Configuration Defines

These are `dwFields` masks indicating NVGVOCONFIG fields to use for `NvGvoGet/Set/Test/CreateDefaultConfig()`.

```
#define NVGVOCONFIG_SIGNALFORMAT      0x00000001   // dwFields: signalFormat
#define NVGVOCONFIG_DATAFORMAT        0x00000002   // dwFields: dataFormat
#define NVGVOCONFIG_OUTPUTREGION      0x00000004   // dwFields: outputRegion
```

```
#define NVGVOCONFIG_OUTPUTAREA        0x00000008   // dwFields: outputArea
#define NVGVOCONFIG_COLORCONVERSION   0x00000010   // dwFields: colorConversion
#define NVGVOCONFIG_GAMMACORRECTION   0x00000020   // dwFields: gammaCorrection
#define NVGVOCONFIG_SYNCSOURCEENABLE  0x00000040   // dwFields: syncSource and
syncEnable
#define NVGVOCONFIG_SYNCDELAY         0x00000080   // dwFields: syncDelay
#define NVGVOCONFIG_COMPOSITESYNCTYPE 0x00000100   // dwFields: compositeSyncType
#define NVGVOCONFIG_FRAMELOCKENABLE   0x00000200   // dwFields: EnableFramelock
#define NVGVOCONFIG_422FILTER         0x00000400   // dwFields: bEnable422Filter
#define NVGVOCONFIG_COMPOSITETERMINATE   0x00000800  // dwFields:
bCompositeTerminate
#define NVGVOCONFIG_DATAINTEGRITYCHECK   0x00001000  // dwFields:
bEnableDataIntegrityCheck
#define NVGVOCONFIG_CSCOVERRIDE      0x00002000 // dwFields: colorConversion
override
#define NVGVOCONFIG_FLIPQUEUELENGTH  0x00004000  // dwFields: flipqueuelength
control
#define NVGVOCONFIG_ANCTIMECODEGENERATION   0x00008000 // dwFields:
bEnableANCTimeCodeGeneration
#define NVGVOCONFIG_COMPOSITE          0x00010000      // dwFields:
bEnableComposite
#define NVGVOCONFIG_ALPHAKEYCOMPOSITE  0x00020000      // dwFields:
bEnableAlphaKeyComposite
#define NVGVOCONFIG_COMPOSITE_Y        0x00040000      // dwFields: compRange
#define NVGVOCONFIG_COMPOSITE_CR       0x00080000      // dwFields: compRange
#define NVGVOCONFIG_COMPOSITE_CB       0x00100000      // dwFields: compRange
#define NVGVOCONFIG_ALLFIELDS   ( NVGVOCONFIG_SIGNALFORMAT          | \
                                  NVGVOCONFIG_DATAFORMAT            | \
                                  NVGVOCONFIG_OUTPUTREGION          | \
                                  NVGVOCONFIG_OUTPUTAREA            | \
                                  NVGVOCONFIG_COLORCONVERSION       | \
                                  NVGVOCONFIG_GAMMACORRECTION       | \
                                  NVGVOCONFIG_SYNCSOURCEENABLE      | \
                                  NVGVOCONFIG_SYNCDELAY             | \
                                  NVGVOCONFIG_COMPOSITESYNCTYPE     | \
                                  NVGVOCONFIG_FRAMELOCKENABLE       | \
                                  NVGVOCONFIG_422FILTER             | \
                                  NVGVOCONFIG_COMPOSITETERMINATE    | \
                                  NVGVOCONFIG_DATAINTEGRITYCHECK    | \
                                  NVGVOCONFIG_CSCOVERRIDE           | \
                                  NVGVOCONFIG_FLIPQUEUELENGTH       | \
                                  NVGVOCONFIG_ANCTIMECODEGENERATION | \
                                  NVGVOCONFIG_COMPOSITE             | \
```

```
                                    NVGVOCONFIG_ALPHAKEYCOMPOSITE     | \
                                    NVGVOCONFIG_COMPOSITE_Y           | \
                                    NVGVOCONFIG_COMPOSITE_CR          | \
                                    NVGVOCONFIG_COMPOSITE_CB)
```

Color Conversion Structure

```
struct  NVGVOCOLORCONVERSION          // Color conversion:
{                                     //
    NVFLOAT       colorMatrix[3][3];  //  Output[n] =
    NVFLOAT       colorOffset[3];     //      Input[0] * colorMatrix[n][0] +
    NVFLOAT       colorScale[3];      //      Input[1] * colorMatrix[n][1] +
                                      //      Input[2] * colorMatrix[n][2] +
                                      //      OutputRange * colorOffset[n]
                                      //   where OutputRange is the standard
                                      //   magnitude of Output[n][n] and
                                       //   colorMatrix and colorOffset values
                                       //   are within the range -1.0 to +1.0
    BOOL          bCompositeSafe;      // bCompositeSafe constrains luminance
                                       // range when using composite output
};
```

Composite Range Structure

```
#define MAX_NUM_COMPOSITE_RANGE    2    // maximum number of ranges per channel
typedef struct tagNVGVOCOMPOSITERANGE
{
    DWORD   dwRange;
    BOOL    bEnabled;
    DWORD   dwMin;
    DWORD   dwMax;
} NVGVOCOMPOSITERANGE;
```

Device Configuration Structure

```
typedef struct tagNVGVOCONFIG
{
    WORD       cbSize;     // Caller sets to sizeof(NVGVOCONFIG)
    DWORD      dwFields;   // Caller sets to NVGVOCONFIG_* mask for fields to use
    NVGVOSIGNALFORMAT signalFormat;     // Signal format for video output
    NVGVODATAFORMAT   dataFormat;       // Data format for video output
    NVGVOOUTPUTREGION outputRegion;     // Region for video output (Desktop mode)
    NVGVOOUTPUTAREA   outputArea;       // Usable resolution for video output
(safe area)
    NVGVOCOLORCONVERSION  colorConversion;    // Color conversion.

    union                           // Gamma correction:
    {                               //  cbSize field in gammaRamp describes type
        NVGAMMARAMP8  gammaRamp8;   //  Gamma ramp (8-bit index, 16-bit values)
        NVGAMMARAMP10 gammaRamp10;  //  Gamma ramp (10-bit index, 16-bit values)
    } gammaCorrection;

    BOOL              syncEnable;   // Sync enable (TRUE to use syncSource)
    NVGVOSYNCSOURCE   syncSource;   // Sync source
    NVGVOSYNCDELAY    syncDelay;    // Sync delay
    NVGVOCOMPSYNCTYPE compositeSyncType;    // Composite sync type
    BOOL     frameLockEnable; // Flag indicating whether framelock was on/off
    double   fGammaValueR;    // Red Gamma value within gamma ranges. 0.5 - 6.0
    double   fGammaValueG;    // Green Gamma value within gamma ranges. 0.5 - 6.0
    double   fGammaValueB;    // Blue Gamma value within gamma ranges. 0.5 - 6.0
    BOOL     bPSFSignalFormat;   // Indicates whether contained format is PSF
Signal format
    BOOL     bEnable422Filter;         // Enables/Disables 4:2:2 filter
    BOOL     bCompositeTerminate;      // Composite termination
    BOOL     bEnableDataIntegrityCheck;  // Enable data integrity check: true -
enable, false - disable
    BOOL     bCSCOverride;       // Use provided CSC color matrix to overwrite
    DWORD    dwFlipQueueLength;          // Number of buffers used for the
internal flipqueue used in pbuffer mode
    BOOL     bEnableANCTimeCodeGeneration; // Enable SDI ANC time code generation
    BOOL     bEnableComposite;             // Enable composite
    BOOL     bEnableAlphaKeyComposite;     // Enable Alpha key composite
    NVGVOCOMPOSITERANGE compRange;         // Composite ranges
    BYTE     reservedData[256];      // Indicates last stored SDI output state
TRUE-ON / FALSE-OFF
} NVGVOCONFIG;
```

Linux CONTROL X Extension API

This section describes the NvGvo APIs inthe following sections:

NV-Control X Functions

Table 5.2 NV-Control X Function Index

Call	Description
XNVCTRLQueryExtension()	Queries for the existence of the Nv_Gvo extensions
XNVCTRLQueryVersion()	Queries the extension version
XNVCTRLIsNvScreen()	Queries whether the specified screen is controlled by the NVIDIA driver.
XNVCTRLSetAttribute()	Sets the specified attribute to the specified value.
XNVCTRLSetAttributeAndGetStatus()	Same as XNVCTRLSetAttribute().
XNVCTRLQueryAttribute()	Queries the value of the specified attribute
XNVCTRLQueryStringAttribute()	Queries the value of the specified string attribute
XNVCTRLSetStringAttribute()	Set the specified string attribute with the specified string.
XNVCTRLQueryValidAttributeValues()	Queries the valid values for the specified attribute
XNVCTRLSetGvoColorConversion()	Sets the color conversion matrix
XNVCTRLQueryGvoColorConversion()	Queries the color conversion matrix

XNVCTRLQueryExtension()

```
Bool XNVCTRLQueryExtension (
    Display *dpy,
    int *event_basep,
    int *error_basep
);
```

This function returns **True** if the extension exists, **False** otherwise. **event_basep** and **error_basep** are the extension event and error bases. Currently, no extension specific errors or events are defined.

XNVCTRLQueryVersion()

```
Bool XNVCTRLQueryVersion (
    Display *dpy,
    int *major,
    int *minor
```

```
);
```

This function returns **True** if the extension exists, **False** otherwise. **major** and **minor** are the extension's major and minor version numbers.

XNVCTRLIsNvScreen()

```
Bool XNVCTRLIsNvScreen (
    Display *dpy,
    int screen
);
```

This function returns **True** is the specified screen is controlled by the NVIDIA driver, otherwise **False**.

XNVCTRLSetAttribute()

```
void XNVCTRLSetAttribute (
    Display *dpy,
    int screen,
    unsigned int display_mask,
    unsigned int attribute,
    int value
);
```

This function sets the attribute to the given value. Not all attributes require the display_mask parameter. See "NV_CTRL_GVO Attributes" on page 75 for details.

Possible errors:

- BadValue - The screen or attribute doesn't exist.
- BadMatch - The NVIDIA driver is not present on that screen.

XNVCTRLSetAttributeAndGetStatus()

```
Bool XNVCTRLSetAttributeAndGetStatus (
    Display *dpy,
    int screen,
    unsigned int display_mask,
    unsigned int attribute,
    int value
```

```
);
```

This function is the same as XNVCTRLSetAttribute(), and returns **True** if the operation succeeds, otherwise **False**.

XNVCTRLQueryAttribute()

```
Bool XNVCTRLQueryAttribute (
    Display *dpy,
    int screen,
    unsigned int display_mask,
    unsigned int attribute,
    int *value
);
```

This function returns **True** if the attribute exists, otherwise **False**.

If XNVCTRLQueryAttribute returns True, value will contain the value of the specified attribute. Not all attributes require the display_mask parameter. See "NV_CTRL_GVO Attributes" on page 75 for details.

Possible errors:

- BadValue - The screen doesn't exist.
- BadMatch - The NVIDIA driver is not present on that screen.

XNVCTRLQueryStringAttribute()

```
Bool XNVCTRLQueryStringAttribute (
    Display *dpy,
    int screen,
    unsigned int display_mask,
    unsigned int attribute,
    char **ptr
);
```

This function returns **True** if the attribute exists, otherwise **False**.

If XNVCTRLQueryStringAttribute returns True, *ptr will point to an allocated string containing the string attribute requested. It is the caller's responsibility to free the string when done.

Possible errors:

- BadValue - The screen doesn't exist.
- BadMatch - The NVIDIA driver is not present on that screen.
- BadAlloc - Insufficient resources to fulfill the request.

XNVCTRLSetStringAttribute()

```
Bool XNVCTRLSetStringAttribute (
    Display *dpy,
    int screen,
    unsigned int display_mask,
    unsigned int attribute,
    char *ptr
);
```

Returns **True** if the operation succeeded, otherwise **False**.

Possible X errors:

- BadValue - The screen doesn't exist.
- BadMatch - The NVIDIA driver is not present on that screen.
- BadAlloc - Insufficient resources to fulfill the request.

XNVCTRLQueryValidAttributeValues()

```
Bool XNVCTRLQueryValidAttributeValues (
    Display *dpy,
    int screen,
    unsigned int display_mask,
    unsigned int attribute,
    NVCTRLAttributeValidValuesRec *values
);
```

This function returns **True** if the attribute exists. otherwise **False**. If XNVCTRLQueryValidAttributeValues returns True, values will indicate the valid values for the specified attribute.

See the description of NVCTRLAttributeValidValues in NVCtrl.h.

XNVCTRLSetGvoColorConversion()

```
void XNVCTRLSetGvoColorConversion (
   Display *dpy,
   int screen,
   float colorMatrix[3][3],
   float colorOffset[3],
   float colorScale[3]
);
```

This function sets the color conversion matrix, offset, and scale that should be used for GVO (Graphic to Video Out).

The Color Space Conversion data is ordered as follows:

- colorMatrix[0][0] // r.Y
- colorMatrix[0][1] // g.Y
- colorMatrix[0][2] // b.Y

- colorMatrix[1][0] // r.Cr
- colorMatrix[1][1] // g.Cr
- colorMatrix[1][2] // b.Cr

- colorMatrix[2][0] // r.Cb
- colorMatrix[2][1] // g.Cb
- colorMatrix[2][2] // b.Cb

- colorOffset[0] // Y
- colorOffset[1] // Cr
- colorOffset[2] // Cb

- colorScale[0] // Y
- colorScale[1] // Cr
- colorScale[2] // Cb

where the data is used according to the following formulae:

- Y = colorOffset[0] + colorScale[0] *
  (R * colorMatrix[0][0] + G * colorMatrix[0][1] + B * colorMatrix[0][2]);

- Cr = colorOffset[1] + colorScale[1] *
  (R * colorMatrix[1][0] + G * colorMatrix[1][1] + B * colorMatrix[1][2]);
- Cb = colorOffset[2] + colorScale[2] *
  (R * colorMatrix[2][0] + G * colorMatrix[2][1] + B * colorMatrix[2][2]);

Possible errors:

- BadMatch - The NVIDIA driver is not present on that screen.
- BadImplementation - GVO is not available on that screen.

XNVCTRLQueryGvoColorConversion()

```
Bool XNVCTRLQueryGvoColorConversion (
    Display *dpy,
    int screen,
    float colorMatrix[3][3],
    float colorOffset[3],
    float colorScale[3]
);
```

This function retrieves the color conversion matrix and color offset that are currently being used for GVO (Graphic to Video Out). The values are ordered within the arrays according to the comments for XNVCTRLSetGvoColorConversion().

Possible errors:

- BadMatch - The NVIDIA driver is not present on that screen.
- BadImplementation - GVO is not available on that screen.

NV_CTRL_GVO Attributes

The NV_CTRL_GVO* integer attributes are used to configure GVO (graphics to video out) functionality on the Quadro FX 4800/5800 SDI graphics board.

The following is a typical usage pattern for the GVO attributes:

- Query NV_CTRL_GVO_SUPPORTED to determine if the X screen supports GV0.
- Specify NV_CTRL_GVO_SYNC_MODE (either FREE_RUNNING, GENLOCK, or

FRAMELOCK).

If you specify GENLOCK or FRAMELOCK, you should also specify NV_CTRL_GVO_SYNC_SOURCE.

- Use NV_CTRL_GVO_SYNC_INPUT_DETECTED and NV_CTRL_GVO_SDI_SYNC_INPUT_DETECTED to detect what input syncs are present.

  If no analog sync is detected but it is known that a valid bi-level or tri-level sync is connected, set NV_CTRL_GVO_COMPOSITE_SYNC_INPUT_DETECT_MODE appropriately and retest with NV_CTRL_GVO_COMPOSITE_SYNC_INPUT_DETECTED.

- If syncing to input sync, query theNV_CTRL_GVO_INPUT_VIDEO_FORMAT attribute.

  The input video format can only be queried after SYNC_SOURCE is specified.

- Specify the NV_CTRL_GVO_OUTPUT_VIDEO_FORMAT.
- Specify the NV_CTRL_GVO_DATA_FORMAT.
- Specify any custom Color Space Conversion (CSC) matrix, offset, and scale with XNVCTRLSetGvoColorConversion().
- If using the GLX_NV_video_out extension to display one or more pbuffers, call glXGetVideoDeviceNV() to lock the GVO output for use by the GLX client, then bind the pbuffer(s) to the GVO output with glXBindVideoImageNV() and send pbuffers to the GVO output with glXSendPbufferToVideoNV().

  See the GLX_NV_video_out spec for more details.

- If, rather than using the GLX_NV_video_out extension to display GLX pbuffers on the GVO output, you wish display the X screen on the GVO output, set NV_CTRL_GVO_DISPLAY_X_SCREEN to NV_CTRL_GVO_DISPLAY_X_SCREEN_ENABLE.

- Setting most GVO attributes only causes the value to be cached in the X server.

  The values will be flushed to the hardware either when NV_CTRL_GVO_DISPLAY_X_SCREEN is enabled, or when a GLX pbuffer is bound to the GVO output (with glXBindVideoImageNV()).

- GLX_NV_video_out and NV_CTRL_GVO_DISPLAY_X_SCREEN are mutually exclusive.

  If NV_CTRL_GVO_DISPLAY_X_SCREEN is enabled, then glXGetVideoDeviceNV will fail. Similarly, if a GLX client has locked the GVO output (via glXGetVideoDeviceNV), then NV_CTRL_GVO_DISPLAY_X_SCREEN will fail. The NV_CTRL_GVO_GLX_LOCKED event will be sent when a GLX client locks the GVO output.

NV_CTRL_GVO_SUPPORTED

```
/*
 * NV_CTRL_GVO_SUPPORTED - returns whether this X screen supports GVO;
 * if this screen does not support GVO output, then all other GVO
 * attributes are unavailable.
 */

#define NV_CTRL_GVO_SUPPORTED                                   67  /* R-- */
#define NV_CTRL_GVO_SUPPORTED_FALSE                             0
#define NV_CTRL_GVO_SUPPORTED_TRUE                              1
```

NV_CTRL_GVO_SYNC_MODE

```
/*
 * NV_CTRL_GVO_SYNC_MODE - selects the GVO sync mode; possible values
 * are:
 *
 * FREE_RUNNING - GVO does not sync to any external signal
 *
 * GENLOCK - the GVO output is genlocked to an incoming sync signal;
 * genlocking locks at hsync.  This requires that the output video
 * format exactly match the incoming sync video format.
 *
 * FRAMELOCK - the GVO output is framelocked to an incoming sync
 * signal; framelocking locks at vsync.  This requires that the output
 * video format have the same refresh rate as the incoming sync video
 * format.
 */

#define NV_CTRL_GVO_SYNC_MODE                                   68  /* RW- */
#define NV_CTRL_GVO_SYNC_MODE_FREE_RUNNING                      0
#define NV_CTRL_GVO_SYNC_MODE_GENLOCK                           1
#define NV_CTRL_GVO_SYNC_MODE_FRAMELOCK                         2
```

NV_CTRL_GVO_SYNC_SOURCE

```
/*
 * NV_CTRL_GVO_SYNC_SOURCE - if NV_CTRL_GVO_SYNC_MODE is set to either
 * GENLOCK or FRAMELOCK, this controls which sync source is used as
```

```
 * the incoming sync signal (either Composite or SDI).  If
 * NV_CTRL_GVO_SYNC_MODE is FREE_RUNNING, this attribute has no
 * effect.
 */

#define NV_CTRL_GVO_SYNC_SOURCE                                  69  /* RW- */
#define NV_CTRL_GVO_SYNC_SOURCE_COMPOSITE                        0
#define NV_CTRL_GVO_SYNC_SOURCE_SDI                              1
```

NV_CTRL_GVO_OUTPUT_VIDEO_FORMAT

```
/*
 * NV_CTRL_GVO_OUTPUT_VIDEO_FORMAT - specifies the output video
 * format.  Note that the valid video formats will vary depending on
 * the NV_CTRL_GVO_SYNC_MODE and the incoming sync video format.  See
 * the definition of NV_CTRL_GVO_SYNC_MODE.
 *
 * Note that when querying the ValidValues for this data type, the
 * values are reported as bits within a bitmask
 * (ATTRIBUTE_TYPE_INT_BITS); unfortunately, there are more valid
 * value bits than will fit in a single 32-bit value.  To solve this,
 * query the ValidValues for NV_CTRL_GVO_OUTPUT_VIDEO_FORMAT to check
 * which of the first 31 VIDEO_FORMATS are valid, then query the
 * ValidValues for NV_CTRL_GVO_OUTPUT_VIDEO_FORMAT2 to check which of
 * the VIDEO_FORMATS with value 32 and higher are valid.
 */

#define NV_CTRL_GVO_OUTPUT_VIDEO_FORMAT                          70  /* RW- */

#define NV_CTRL_GVO_VIDEO_FORMAT_NONE                            0
#define NV_CTRL_GVO_VIDEO_FORMAT_480I_59_94_SMPTE259_NTSC        1
#define NV_CTRL_GVO_VIDEO_FORMAT_576I_50_00_SMPTE259_PAL         2
#define NV_CTRL_GVO_VIDEO_FORMAT_720P_59_94_SMPTE296             3
#define NV_CTRL_GVO_VIDEO_FORMAT_720P_60_00_SMPTE296             4
#define NV_CTRL_GVO_VIDEO_FORMAT_1035I_59_94_SMPTE260            5
#define NV_CTRL_GVO_VIDEO_FORMAT_1035I_60_00_SMPTE260            6
#define NV_CTRL_GVO_VIDEO_FORMAT_1080I_50_00_SMPTE295            7
#define NV_CTRL_GVO_VIDEO_FORMAT_1080I_50_00_SMPTE274            8
#define NV_CTRL_GVO_VIDEO_FORMAT_1080I_59_94_SMPTE274            9
#define NV_CTRL_GVO_VIDEO_FORMAT_1080I_60_00_SMPTE274            10
```

```
#define NV_CTRL_GVO_VIDEO_FORMAT_1080P_23_976_SMPTE274          11
#define NV_CTRL_GVO_VIDEO_FORMAT_1080P_24_00_SMPTE274           12
#define NV_CTRL_GVO_VIDEO_FORMAT_1080P_25_00_SMPTE274           13
#define NV_CTRL_GVO_VIDEO_FORMAT_1080P_29_97_SMPTE274           14
#define NV_CTRL_GVO_VIDEO_FORMAT_1080P_30_00_SMPTE274           15
#define NV_CTRL_GVO_VIDEO_FORMAT_720P_50_00_SMPTE296            16
#define NV_CTRL_GVO_VIDEO_FORMAT_1080I_24_00_SMPTE274           17 //deprecated
#define NV_CTRL_GVO_VIDEO_FORMAT_1080I_48_00_SMPTE274           17
#define NV_CTRL_GVO_VIDEO_FORMAT_1080I_23_98_SMPTE274           18 //deprecated
#define NV_CTRL_GVO_VIDEO_FORMAT_1080I_47_96_SMPTE274           18
#define NV_CTRL_GVO_VIDEO_FORMAT_720P_30_00_SMPTE296            19
#define NV_CTRL_GVO_VIDEO_FORMAT_720P_29_97_SMPTE296            20
#define NV_CTRL_GVO_VIDEO_FORMAT_720P_25_00_SMPTE296            21
#define NV_CTRL_GVO_VIDEO_FORMAT_720P_24_00_SMPTE296            22
#define NV_CTRL_GVO_VIDEO_FORMAT_720P_23_98_SMPTE296            23
#define NV_CTRL_GVO_VIDEO_FORMAT_1080PSF_25_00_SMPTE274         24
#define NV_CTRL_GVO_VIDEO_FORMAT_1080PSF_29_97_SMPTE274         25
#define NV_CTRL_GVO_VIDEO_FORMAT_1080PSF_30_00_SMPTE274         26
#define NV_CTRL_GVO_VIDEO_FORMAT_1080PSF_24_00_SMPTE274         27
#define NV_CTRL_GVO_VIDEO_FORMAT_1080PSF_23_98_SMPTE274         28
#define NV_CTRL_GVO_VIDEO_FORMAT_1080P_30_00_SMPTE372           29
#define NV_CTRL_GVO_VIDEO_FORMAT_1080P_29_97_SMPTE372           30
#define NV_CTRL_GVO_VIDEO_FORMAT_1080I_30_00_SMPTE372           31
#define NV_CTRL_GVO_VIDEO_FORMAT_1080I_29_97_SMPTE372           32
#define NV_CTRL_GVO_VIDEO_FORMAT_1080P_25_00_SMPTE372           33
#define NV_CTRL_GVO_VIDEO_FORMAT_1080I_25_00_SMPTE372           34
#define NV_CTRL_GVO_VIDEO_FORMAT_1080P_24_00_SMPTE372           35
#define NV_CTRL_GVO_VIDEO_FORMAT_1080P_23_98_SMPTE372           36
#define NV_CTRL_GVO_VIDEO_FORMAT_1080I_24_00_SMPTE372           37
#define NV_CTRL_GVO_VIDEO_FORMAT_1080I_23_98_SMPTE372           38
```

NV_CTRL_GVO_INPUT_VIDEO_FORMAT

```
/*
 * NV_CTRL_GVO_INPUT_VIDEO_FORMAT - indicates the input video format
 * detected; the possible values are the NV_CTRL_GVO_VIDEO_FORMAT
 * constants.
 */

#define NV_CTRL_GVO_INPUT_VIDEO_FORMAT                          71  /* R-- */
```

NV_CTRL_GVO_DATA_FORMAT

```
/*
 * NV_CTRL_GVO_DATA_FORMAT - This controls how the data in the source
 * (either the X screen or the GLX pbuffer) is interpretted and
 * displayed.
 */

#define NV_CTRL_GVO_DATA_FORMAT                                    72  /* RW- */
#define NV_CTRL_GVO_DATA_FORMAT_R8G8B8_TO_YCRCB444                 0
#define NV_CTRL_GVO_DATA_FORMAT_R8G8B8A8_TO_YCRCBA4444             1
#define NV_CTRL_GVO_DATA_FORMAT_R8G8B8Z10_TO_YCRCBZ4444            2
#define NV_CTRL_GVO_DATA_FORMAT_R8G8B8_TO_YCRCB422                 3
#define NV_CTRL_GVO_DATA_FORMAT_R8G8B8A8_TO_YCRCBA4224             4
#define NV_CTRL_GVO_DATA_FORMAT_R8G8B8Z10_TO_YCRCBZ4224            5
#define NV_CTRL_GVO_DATA_FORMAT_R8G8B8_TO_RGB444                   6
#define NV_CTRL_GVO_DATA_FORMAT_R8G8B8A8_TO_RGBA4444               7
#define NV_CTRL_GVO_DATA_FORMAT_R8G8B8Z10_TO_RGBZ4444              8
#define NV_CTRL_GVO_DATA_FORMAT_Y10CR10CB10_TO_YCRCB444            9
#define NV_CTRL_GVO_DATA_FORMAT_Y10CR8CB8_TO_YCRCB444              10
#define NV_CTRL_GVO_DATA_FORMAT_Y10CR8CB8A10_TO_YCRCBA4444         11
#define NV_CTRL_GVO_DATA_FORMAT_Y10CR8CB8Z10_TO_YCRCBZ4444         12
#define NV_CTRL_GVO_DATA_FORMAT_DUAL_R8G8B8_TO_DUAL_YCRCB422       13
#define NV_CTRL_GVO_DATA_FORMAT_DUAL_Y8CR8CB8_TO_DUAL_YCRCB422     14
#define NV_CTRL_GVO_DATA_FORMAT_R10G10B10_TO_YCRCB422              15
#define NV_CTRL_GVO_DATA_FORMAT_R10G10B10_TO_YCRCB444              16
#define NV_CTRL_GVO_DATA_FORMAT_Y12CR12CB12_TO_YCRCB444            17
#define NV_CTRL_GVO_DATA_FORMAT_R12G12B12_TO_YCRCB444              18
```

NV_CTRL_GVO_DISPLAY_X_SCREEN

```
/*
 * NV_CTRL_GVO_DISPLAY_X_SCREEN - enable/disable GVO output of the X
 * screen.  At this point, all the GVO attributes that have been
 * cached in the X server are flushed to the hardware and GVO is
 * enabled.  Note that this attribute can fail to be set if a GLX
 * client has locked the GVO output (via glXGetVideoDeviceNV).  Note
 * that due to the inherit race conditions in this locking strategy,
 * NV_CTRL_GVO_DISPLAY_X_SCREEN can fail unexpectantly.  In the
 * failing situation, X will not return an X error.  Instead, you
```

```
 * should query the value of NV_CTRL_GVO_DISPLAY_X_SCREEN after
 * setting it to confirm that the setting was applied.
 */

#define NV_CTRL_GVO_DISPLAY_X_SCREEN                            73  /* RW- */
#define NV_CTRL_GVO_DISPLAY_X_SCREEN_ENABLE                     1
#define NV_CTRL_GVO_DISPLAY_X_SCREEN_DISABLE                    0
```

NV_CTRL_GVO_COMPOSITE_SYNC_INPUT_DETECTED

```
/*
 * NV_CTRL_GVO_COMPOSITE_SYNC_INPUT_DETECTED - indicates whether
 * Composite Sync input is detected.
 */

#define NV_CTRL_GVO_COMPOSITE_SYNC_INPUT_DETECTED               74  /* R-- */
#define NV_CTRL_GVO_COMPOSITE_SYNC_INPUT_DETECTED_FALSE         0
#define NV_CTRL_GVO_COMPOSITE_SYNC_INPUT_DETECTED_TRUE          1
```

NV_CTRL_GVO_COMPOSITE_SYNC_INPUT_DETECT_MODE

```
/*
 * NV_CTRL_GVO_COMPOSITE_SYNC_INPUT_DETECT_MODE - get/set the
 * Composite Sync input detect mode.
 */

#define NV_CTRL_GVO_COMPOSITE_SYNC_INPUT_DETECT_MODE            75  /* RW- */
#define NV_CTRL_GVO_COMPOSITE_SYNC_INPUT_DETECT_MODE_AUTO       0
#define NV_CTRL_GVO_COMPOSITE_SYNC_INPUT_DETECT_MODE_BI_LEVEL   1
#define NV_CTRL_GVO_COMPOSITE_SYNC_INPUT_DETECT_MODE_TRI_LEVEL  2
```

NV_CTRL_GVO_SYNC_INPUT_DETECTED

```
/*
 * NV_CTRL_GVO_SYNC_INPUT_DETECTED - indicates whether SDI Sync input
 * is detected, and what type.
 */

#define NV_CTRL_GVO_SDI_SYNC_INPUT_DETECTED                     76  /* R-- */
```

```
#define NV_CTRL_GVO_SDI_SYNC_INPUT_DETECTED_NONE                0
#define NV_CTRL_GVO_SDI_SYNC_INPUT_DETECTED_HD                  1
#define NV_CTRL_GVO_SDI_SYNC_INPUT_DETECTED_SD                  2
```

NV_CTRL_GVO_VIDEO_OUTPUTS

```
/*
 * NV_CTRL_GVO_VIDEO_OUTPUTS - indicates which GVO video output
 * connectors are currently outputing data.
 */

#define NV_CTRL_GVO_VIDEO_OUTPUTS                               77  /* R-- */
#define NV_CTRL_GVO_VIDEO_OUTPUTS_NONE                          0
#define NV_CTRL_GVO_VIDEO_OUTPUTS_VIDEO1                        1
#define NV_CTRL_GVO_VIDEO_OUTPUTS_VIDEO2                        2
#define NV_CTRL_GVO_VIDEO_OUTPUTS_VIDEO_BOTH                    3
```

NV_CTRL_GVO_FPGA_VERSION

```
/*
 * NV_CTRL_GVO_FPGA_VERSION - indicates the version of the Firmware on
 * the GVO device.  XXX would this be better as a string attribute?
 */

#define NV_CTRL_GVO_FIRMWARE_VERSION                            78  /* R-- */
```

NV_CTRL_GVO_SYNC_DELAY_PIXELS

```
/*
 * NV_CTRL_GVO_SYNC_DELAY_PIXELS - controls the delay between the
 * input sync and the output sync in numbers of pixels from hsync;
 * this is a 12 bit value.
 */

#define NV_CTRL_GVO_SYNC_DELAY_PIXELS                           79  /* RW- */
```

NV_CTRL_GVO_SYNC_DELAY_LINES

```
/*
 * NV_CTRL_GVO_SYNC_DELAY_LINES - controls the delay between the input
 * sync and the output sync in numbers of lines from vsync; this is a
 * 12 bit value.
 */

#define NV_CTRL_GVO_SYNC_DELAY_LINES                            80  /* RW- */
```

NV_CTRL_GVO_INPUT_VIDEO_FORMAT_REACQUIRE

```
/*
 * NV_CTRL_GVO_INPUT_VIDEO_FORMAT_REACQUIRE - must be set for a period
 * of about 2 seconds for the new InputVideoFormat to be properly
 * locked to.  In nvidia-settings, we do a reacquire whenever genlock
 * or framelock mode is entered into, when the user clicks the
 * "detect" button.  This value can be written, but always reads back
 * _FALSE.
 */

#define NV_CTRL_GVO_INPUT_VIDEO_FORMAT_REACQUIRE                81  /* -W- */
#define NV_CTRL_GVO_INPUT_VIDEO_FORMAT_REACQUIRE_FALSE          0
#define NV_CTRL_GVO_INPUT_VIDEO_FORMAT_REACQUIRE_TRUE           1
```

NV_CTRL_GVO_GLX_LOCKED

```
/*
 * NV_CTRL_GVO_GLX_LOCKED - indicates that GVO configurability is locked by
 * GLX;  this occurs when the GLX_NV_video_out function calls
 * glXGetVideoDeviceNV().  All GVO output resources are locked until
 * either glXReleaseVideoDeviceNV() is called or the X Display used
 * when calling glXGetVideoDeviceNV() is closed.
 *
 * When GVO is locked, setting of the following GVO NV-CONTROL attributes will
 * not happen immediately and will instead be cached.  The GVO resource will
 * need to be disabled/released and re-enabled/claimed for the values to be
 * flushed. These attributes are:
 *    NV_CTRL_GVO_OUTPUT_VIDEO_FORMAT
 *    NV_CTRL_GVO_DATA_FORMAT
```

```
 *    NV_CTRL_GVO_FLIP_QUEUE_SIZE
 *
 * XXX This is deprecated, please see NV_CTRL_GVO_LOCK_OWNER
 */

#define NV_CTRL_GVO_GLX_LOCKED                                  82  /* R-- */
#define NV_CTRL_GVO_GLX_LOCKED_FALSE                            0
```

#define NV_CTRL_GVO_GLX_LOCKED_TRUE 1

NV_CTRL_GVO_VIDEO_FORMAT_{WIDTH,HEIGHT,REFRESH_RATE}

```
/*
 * NV_CTRL_GVO_VIDEO_FORMAT_{WIDTH,HEIGHT,REFRESH_RATE} - query the
 * width, height, and refresh rate for the specified
 * NV_CTRL_GVO_VIDEO_FORMAT_*.  So that this can be queried with
 * existing interfaces, XNVCTRLQueryAttribute() should be used, and
 * the video format specified in the display_mask field; eg:
 *
 * XNVCTRLQueryAttribute (dpy,
 *                        screen,
 *                        NV_CTRL_GVO_VIDEO_FORMAT_480I_59_94_SMPTE259_NTSC
 *                        NV_CTRL_GVO_VIDEO_FORMAT_WIDTH,
 *                        &value);
 *
 * Note that Refresh Rate is in 1/1000 Hertz values
 */

#define NV_CTRL_GVO_VIDEO_FORMAT_WIDTH                          83  /* R-- */
#define NV_CTRL_GVO_VIDEO_FORMAT_HEIGHT                         84  /* R-- */
#define NV_CTRL_GVO_VIDEO_FORMAT_REFRESH_RATE                   85  /* R-- */
```

NV_CTRL_GVO_X_SCREEN_PAN_[XY]

```
/*
 * NV_CTRL_GVO_X_SCREEN_PAN_[XY] - when GVO output of the X screen is
 * enabled, the pan x/y attributes control which portion of the X
 * screen is displayed by GVO.  These attributes can be updated while
 * GVO output is enabled, or before enabling GVO output.  The pan
 * values will be clamped so that GVO output is not panned beyond the
```

```
 * end of the X screen.
 */

#define NV_CTRL_GVO_X_SCREEN_PAN_X                      86  /* RW- */
#define NV_CTRL_GVO_X_SCREEN_PAN_Y                      87  /* RW- */
```

APPENDIX

A

ONBOARD DIP SWITCH

The Quadro FX SDI graphics card has an onboard dip switch, located on the SDI output card, that determines the default SDI operating mode. Subsequent software changes override these settings.

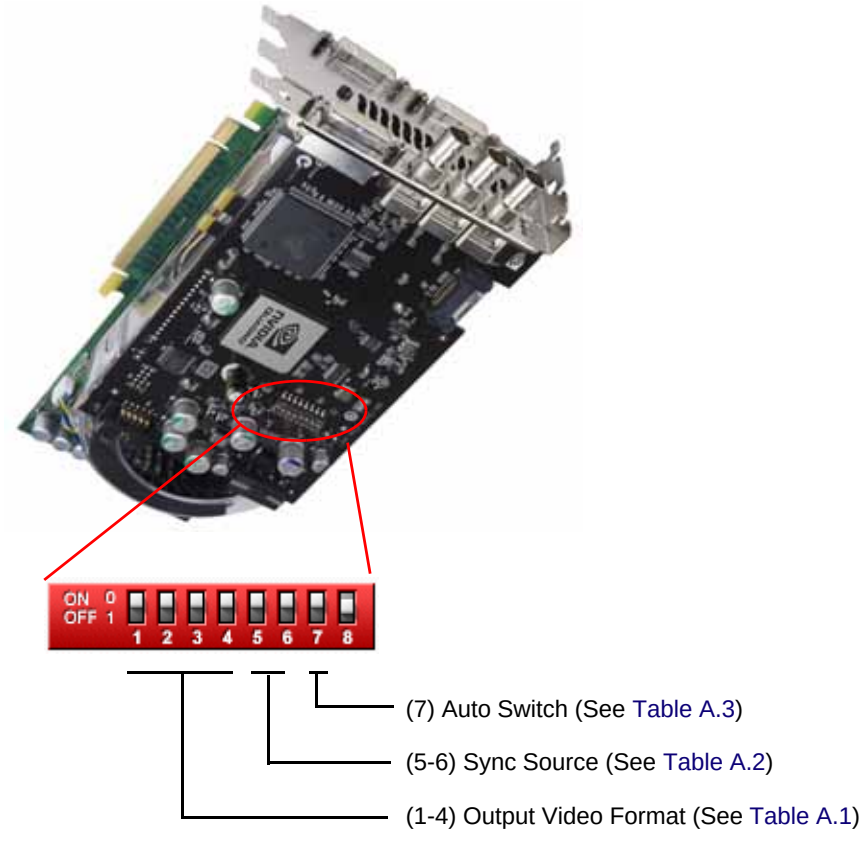


Figure 1.1 Onboard DIP Switch Positions

In the following tables, a “0” value corresponds to the “ON” switch position, and a “1” value corresponds to the “OFF” switch position.

Table A.1 Output Video Format Switch Settings

Switch Position	
1234	**Video Format**
0000	Reserved
1000	SMPTE 259 NTSC, 1440x487, 30/1.001 Hz, Interlace
0100	SMPTE 259 PAL, 1440x576, 25 Hz, Interlace
1100	SMPTE 260, 1920x1035, 30 Hz, Interlace
0010	SMPTE 260, 1920x1035, 30/1.001 Hz, Interlace
1010	SMPTE 295, 1920x1080, 25 Hz, Interlace
0110	SMPTE 274, 1920x1080, 30 Hz, Interlace
1110	SMPTE 274, 1920x1080, 30/1.001 Hz, Interlace
0001	SMPTE 274, 1920x1080, 25 Hz, Interlace
1001	SMPTE 274, 1920x1080, 30 Hz, Progressive
0101	SMPTE 274, 1920x1080, 30/1.001 Hz, Progressive
1101	SMPTE 274, 1920x1080, 25 Hz, Progressive
0011	SMPTE 274, 1920x1080, 24 Hz, Progressive
1011	SMPTE 274, 1920x1080, 24/1.001 Hz, Progressive
0111	SMPTE 296, 1280x720, 60 Hz, Progressive
1111	SMPTE 296, 1280x720, 60/1.001 Hz, Progressive

Table A.2 Sync Source Switch Settings

Switch Position	
56	**Sync Source**
00	Internal (free running)
10	Synchronize to SDI sync source
01	Synchronize to Composite sync source
11	Reserved

Table A.3 Auto Switch Settings

Switch Position	
7	**Auto Switch Setting**
0	Do not auto switch
1	Automatically switch to the new video format based on the source sync.