Table Of Contents

Quick Install

For those anxious to get started, this describes a succinct process for installing the NVR Development kit and starting it up.

Briefly put, you insert the release tape in the tape drive, make a directory for the distribution, start the extraction from tape, then go have a cup of coffee. When you get back, set up the environment and you are ready to go!

Before you start you need a few items:

What you need to start

- A Sun SPARCstation 2, IPC, IPX or anything faster, running SunOS 4.1.x or Solaris 1.0.1. If it has a Parallax XVideo frame-buffer or a standard Sun frame buffer, then the video decoder will run at top speed. We suggest OpenWindows 3.0.
- An SGI workstation running IRIX 5.0.2 or greater.
- 80 megabytes of free space. This is how much space is taken by the programs and the compressed sequences.
- A tape drive appropriate for the distribution media you recieved, typically an 8mm tape or a standard quarter-inch cartridge (QIC).

Installing the software

Now, assuming you are familiar with the UNIX™ C-shell, enter the following commands:

```
% mkdir nvr
% cd nvr
% tar xf /dev/rst0
```

Now go have a cup of coffee. When you return and the tape has rewound, you should

- setup the NVRHOME environment variable
- change your path to include `nvr/bin`
- change your search path for X resource application defaults.

Do this as follows:

```
% setenv NVRHOME $cwd
% set path = ($NVRHOME/bin $NVRHOME/adm/bin $path)
% setenv XFILESEARCHPATH \
      $NVRHOME/Xdata/app-defaults/%N%S
```

Introduction

Congratulations! You now posses a well-integrated system for recording (on some systems), perusing and compressing video and audio under UNIX and the X Window System™.

This set of products from NVR is sold as a kit or separately. The full kit current consists of three Motif™ user interfaces:

- The Player. This tool provides a viewing mechanism for working with analog video (which requires a Parallax Xvideo board), JPEG movies (which requires the Parallax Xvideo board with JPEG), and MPEG.
- The Recorder. This tool enables the user to peruse analog material with an interface very similar to the Player, but in addition, allows you to record video using the JPEG hardware on the parallax board. Sequences may be “running records” from a camera or broadcast, or assemble editing with in and out points. Both the Player and Recorder support Sony’s ViSCA/LANC, and the Pioneer 4400 video disc player[1]. VideoMedia’s VLAN will be added soon.
- The Compressor. This tool allows you to gather up input files and compress them with our MPEG compression engine. We support variable I:P:B ratios, variable a fixed bit rate, and three motion estimation algorithms (Jain & Jain and two Exhaustive methods).

In addition, the audio compressor, system multiplexer and other tools can be used in a command-environment.

The MPEG playback mechanism consists purely of software, requires no special framebuffer, and depending on the size of picture and bandwidth of the bitstream, can run at 6 - 24 fps with synchronized audio.

The color in MPEG playback is dithered from 19 bits down to 7 bits, gamma-corrected, and has real-time adjustments for contrast and brightness.

We are very anxious to make our software systems the industry standard for video under UNIX and X, so we welcome any and all com-

1. Other computer-controllable Pioneer laser video disc players may work with NVR software as well but have not been tested.

What Is Digitized Video?

Digitized video is new-age analog video.

There was a time, recently in fact, when the term "digital video" referred to the two digital tape formats used in the highest quality video post-production studios: D1 and D2. Times have changed and the popular computing movement has usurped the term to describe digitized video put to use in computers. Redefining the term wasn't a calculated move, but rather a result of the immense number of computers that are now fast enough to support some form of motion picture playback and the even larger number of people that use the computers.

Still the majority of computers have a long way to go, partly because workstations and personal computers (PCs) only became fast enough around 1989 or 1990, and partly because most software systems are not designed with digital video in mind. The speed of computers is a barrier to digital video on all but these most recent computers, and the design of software systems limits that group even further.

This section is intended to inform you about the processes involved in transforming analog video into digital video. There's nothing you or I can do to make computers go faster ... the computer vendors will continue to make that happen without our help[1]. Concerning the software, we think that you will begin to draw the same conclusions about software systems as you read this and understand more about the video digitization process.

1. Some in the industry say that computers, starting about 1980, double in speed every two years. It hasn't stopped yet.

Window System to display a series of images is difficult because X was designed for discrete graphic requests like “draw a line” or “fill a rectangle”, not continuous media. Nevertheless, it is possible, and using the X extension mechanism to get around the window system, workstation and frame buffer vendors are able to see reasonable performance.

Frame Buffer Bandwidth

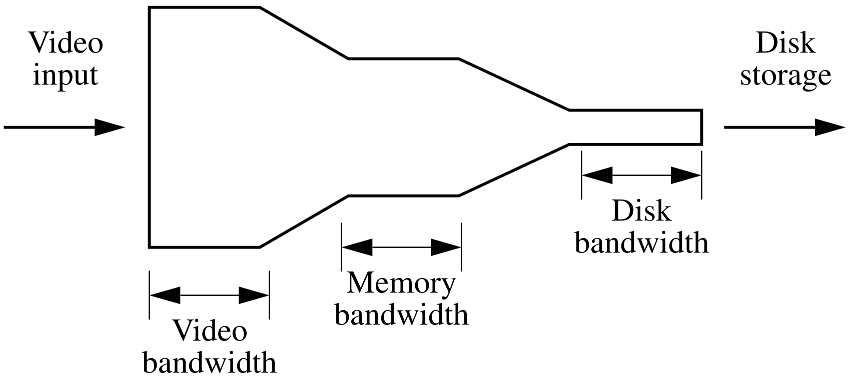
Some tangible numbers: a Sun SPARCstation 2 can usually do about 5 megatransfers[1] per second directly to the frame buffer. That’s about the best. The worst is with a vanilla X server using a socket mechanism, which is limited by a number of factors, but 1 MB/sec is a common figure.

System Memory

Memory read/write access on the SPARCstation 2 is about 350 nanoseconds (ns) per longword, or 2.9 megatransfers per second, or 11.5MB/sec. The write-through cache architecture means that memory-writes never perform any better than this. Memory reads can go much faster if they fall in the cache: about 50ns per longword or 80MB/sec.

Add It All UP

These are many numbers to juggle, but we read them this way: To capture video and store it on your workstation, it must first be reduced enough for the processor to copy it into memory, and then reduced further to copy it onto the disk. Note that each reduction step may take no time (if its done by hardware or is a simple software task), or a lot of time (software compression). Once it is on the disk, the expansion is limited by the same type of rules.



1. A “transfer” is a read or write of any size: 1 byte, 2 bytes or 4 bytes, it matters not.

depth	width	height	frame rate
32 bits	122	92	10 fps
32 bits	172	129	5 fps
16 bits	345	259	5 fps

Software Compression and Decompression

Once it has been captured at the best quality, you can decide on the quality and frame rate requirements of your application. Applying MPEG compression can result in better quality than JPEG for a given size and frame rate. MPEG works well in a wide range of frame rates and bit rates. We find that 50 - 200 KB/sec allows it to be stored on and retrieved from any disk.

Summary

We think that because of the immense data reduction needed and a desire for quality, hardware compression as a first step in working with digital video is the only way to go. After that MPEG compression can be used to achieve the desired frame rate and bandwidth, and our MPEG software decompression allows it to be accessible on any platform.

How To View Sequences

This section is a tutorial about how to view digital or analog video sequences using NVR's `player` program. We make the following assumptions about what you know and have done before reading this:

- You have a basic understanding of the Motif™ user interface styles; that is, you can find your way around a Motif environment pretty well. (Check your local technical bookstore for Motif user interface books)
- You have installed the NVR Development Kit. (See Quick Install on page x or Installation Guide in Appending C.
- You have added `nvr/bin` to your path

Starting the Player

To start the player execute the following command:

```
% player -path $NVRHOME/sequences/mpeg &
```

By default the player comes up in MPEG mode. On some systems the player is actually three applications in one:

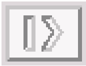
- MPEG player
  Plays back MPEG compressed digital video and audio files.
- JPEG player
  Plays back JPEG compressed digital video and audio files.
- Analog Video player
  Plays video and audio from analog video devices.

You can bring up the JPEG and Analog player on startup via the *-jpeg* or *-analog* command line option, but for this tutorial let's just start with MPEG. We'll be dynamically changing between the 3 interfaces via a pulldown menu.

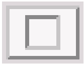
MPEG Player

To pause a sequence that is in play, simply press the **Pause** button

To single step forward through frames press the **Single-Step-Forward** button



To stop and rewind a sequence to its beginning, press the **stop** button

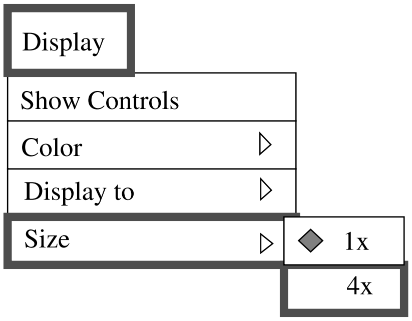


Go ahead and experiment with these buttons.

When you’re done experimenting with the playback controls, let’s look at the MPEG player’s display options. But before proceeding let’s insure we are at the same point by pressing the **Stop** button.

Screen Size

The MPEG player can render frames 4 times its regular size by replicating pixels. To enable this feature, press on the **Display** button in the menubar. This will “pulldown” the Display Menu. Now press on the **Size** button, then the **4x** button from the resulting cascade menu as shown below.



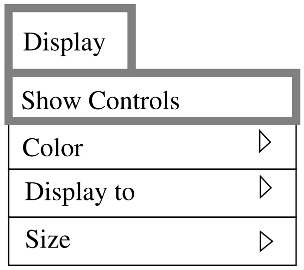
Now redisplay the current frame by pressing **Single-Step-Forward** button or if you just want to play the sequence pressing the **Play** button.

When you’re done experimenting with this feature, revert back to the 1x mode by again pulling down the Display Menu and pressing the **Size** button but this time press the **1x** button from the cascade menu.

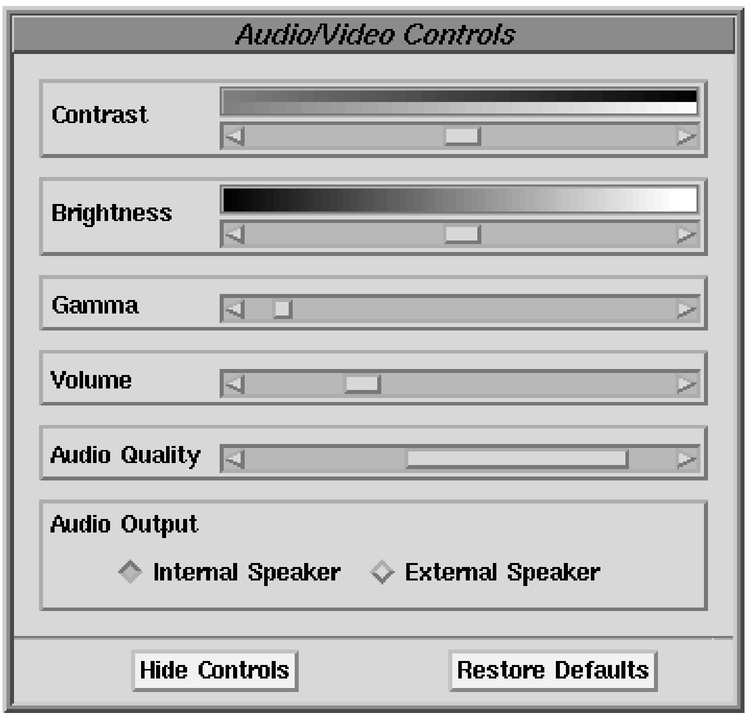
Play button.

Audio and Video Controls

The next group of display controls can be manipulated while the sequence is playing. Let's bring up that control panel right now by pressing the **Show Controls** button from the Display Menu as shown below.



The Audio/Video Controls panel as shown below should appear.

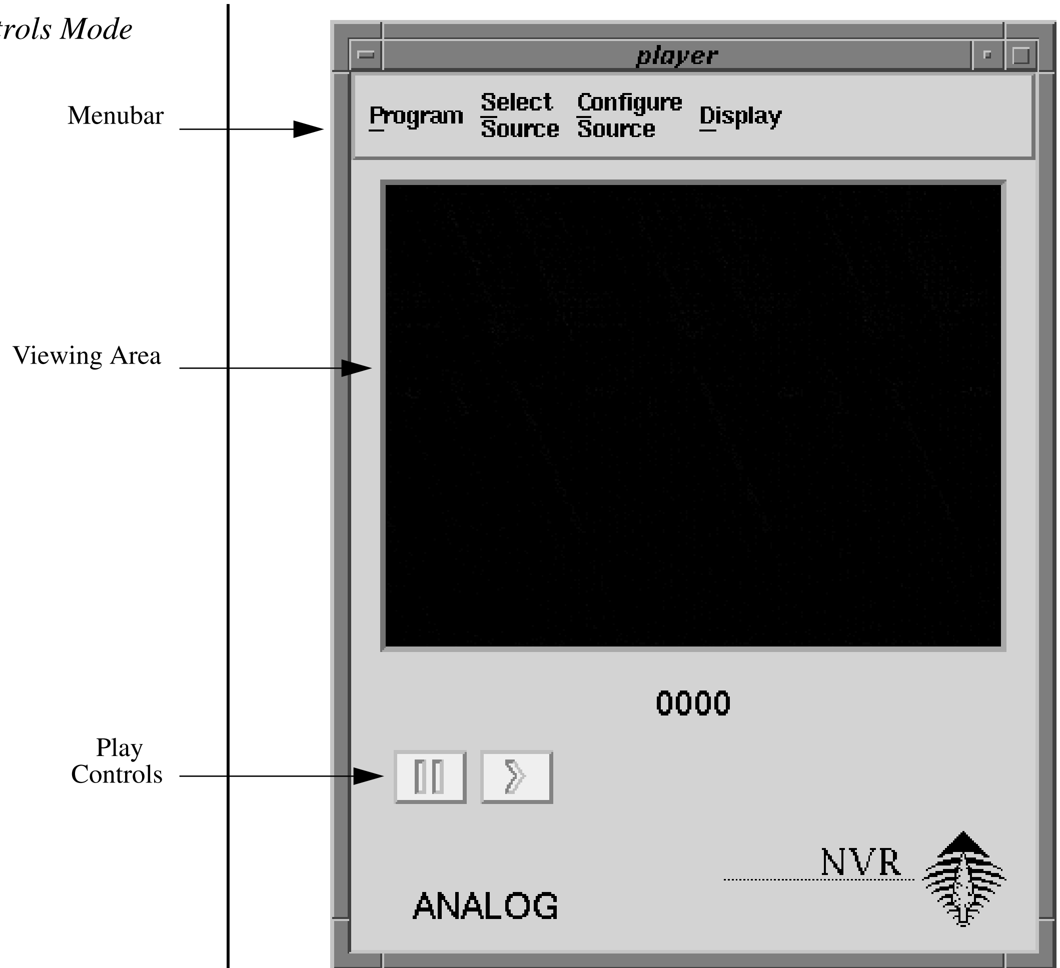


Let's experiment with these sliders and toggles, but first change our display mode back to color by bringing up the Display Menu, then pressing the **Color** buttons.

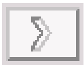
At this time play the MPEG sequence, but while its playing experiment with the **Contrast**, **Brightness**, and **Gamma** sliders.

instructions at the end of this subsection.

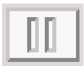
No-Controls Mode



By default the Analog Video Player interface comes up in its simplest configuration as shown above. In this mode the player has a video display pane acting as a passive video monitor. To activate the video display press the Play button.

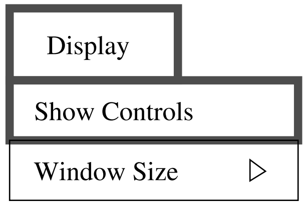


To freeze the picture in the display, press the Pause button.

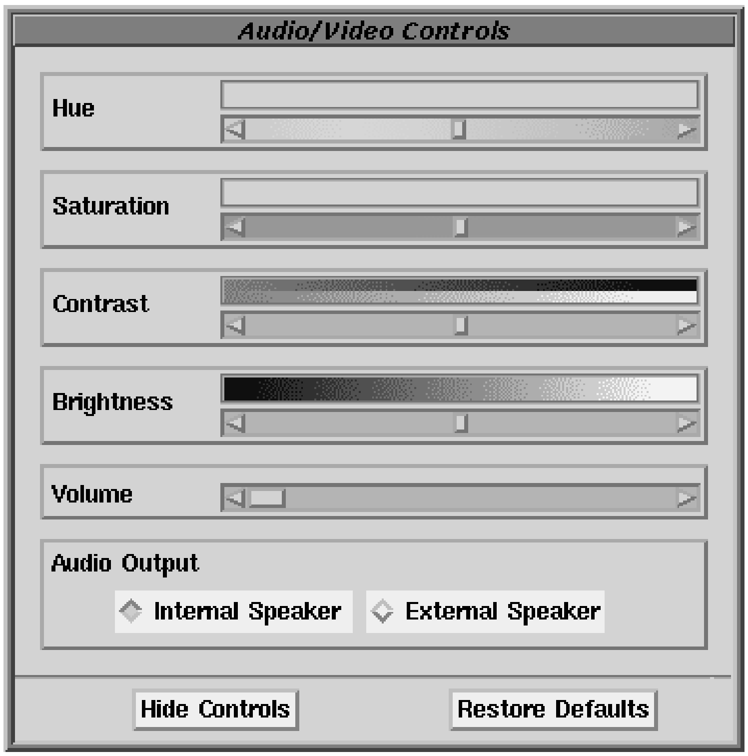


In this no-controls mode, if you have the video display active and your video source device is properly connected to the Parallax, you manually control your source device to display an image in the video display.

play item from the menu bar and select **Show Display Controls**.



The Analog Video Audio/Video Control panel containing selections to adjustments to the video image being displayed should appear as shown below.

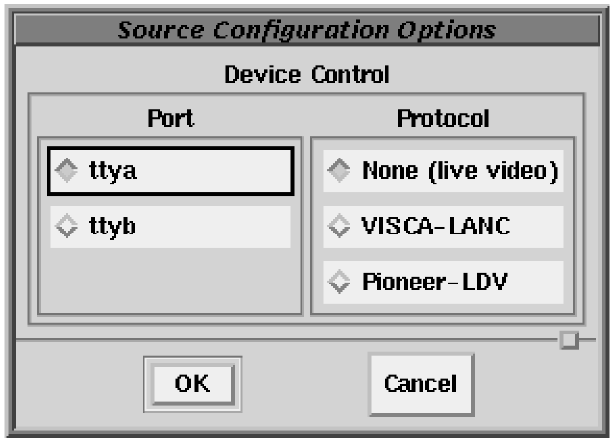


The difference with this Audio/Video Controls panel and the MPEG Player's Audio/Video Controls panel is the addition of the **Hue** and **Saturation** sliders; and the elimination of the **Audio Quality** slider.

You can use these controls to improve the appearance of the analog video and, as with the MPEG Player, control the audio volume and audio output.

Window Size

The Analog Video Player can display video in either 320x240 or



You'll need to specify two pieces of information. One is the RS-232 port connected to the device (either ttya or ttyb). The other is whether you are connecting a Pioneer LD-V video disk player or a VISCA/LANC-controllable device connected through a Vbox. Once you have made your selections, press the **OK** button to have them take effect. Additional control buttons will appear in the main window of the Analog Video Player. Should you decide not to add or change device control, press the **Cancel** button to close the dialog with no effect.

.

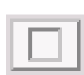
The additional control buttons are:

 Fast-Reverse

 Single-Step-Forward

 Single-Step-Reverse

 Fast-Forward

 Stop

File Selection

File selection is performed with the Motif file selection dialog. To bring up the file selection dialog for JPEG files, press and release the **Select File** button. Now you can peruse through the file system and select your desired file, however, for the purpose of this tutorial select the file "`solder.jpeg`".

Playback Controls

By now, the first frame of the JPEG sequences should be displayed. Control buttons available in the JPEG Player are:

 Fast-Reverse

 Single-Step-Reverse

 Pause

 Play

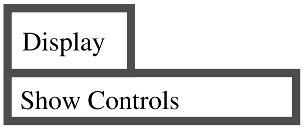
 Single-Step-Forward

 Fast-Forward

To play, simply press the **Play** button. Go ahead and experiment with the all these buttons.

Audio Controls

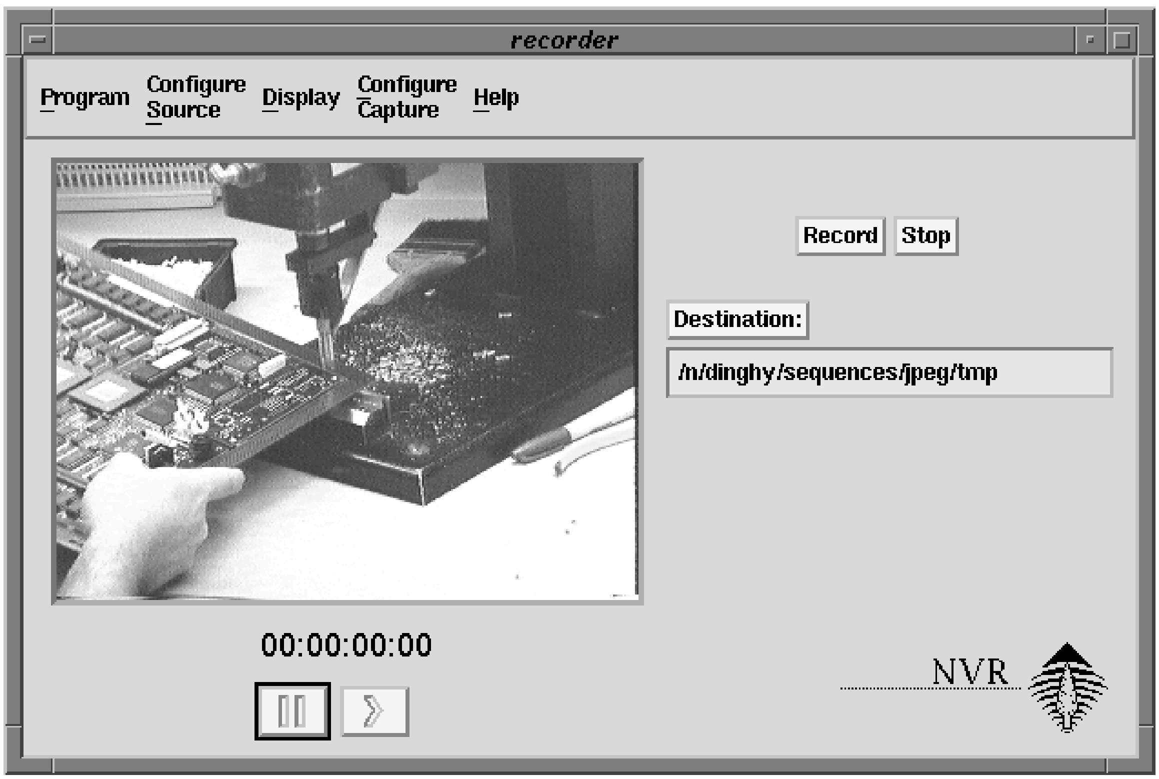
When you're done experimenting with the playback controls, let's look at the JPEG player's display controls. To bring up the control panel press the **Show Controls** button from the Display Menu as shown below.



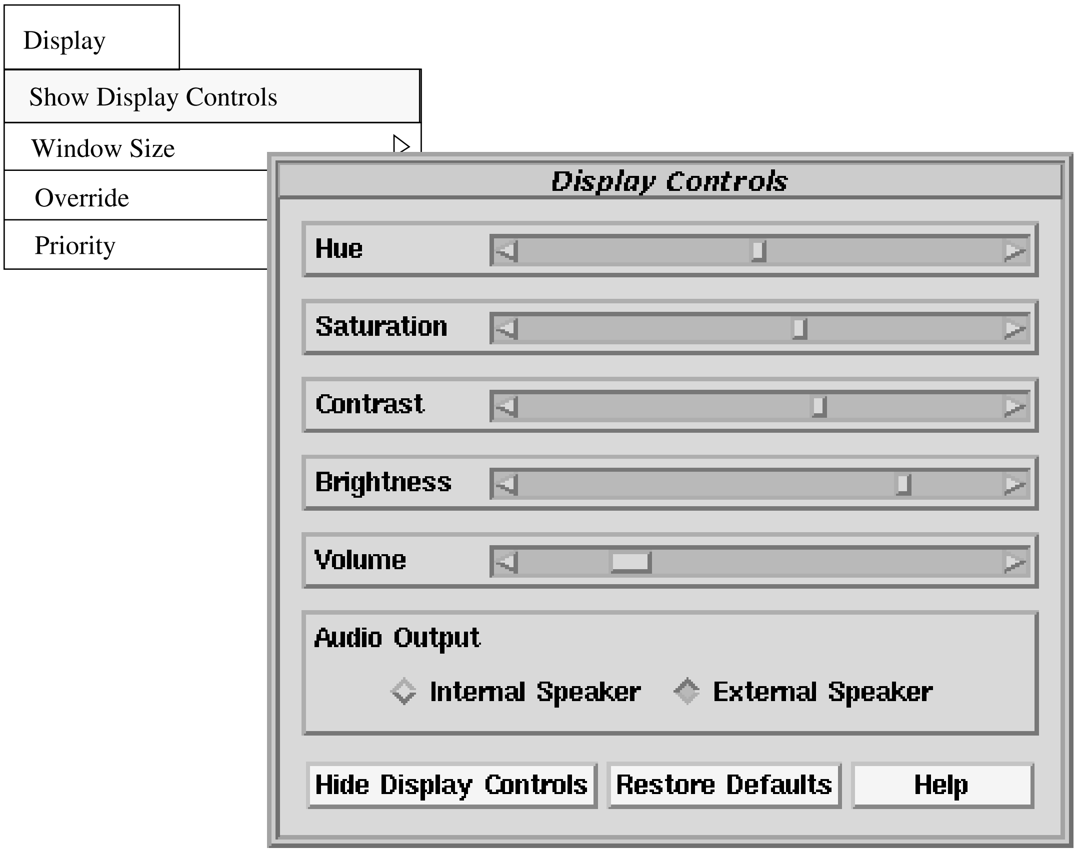
How To Record Sequences

This section is a tutorial about how to record sequences from analog source devices into JPEG movie files using NVR's `recorder` application program. We make the following assumptions about what you know and have done before reading this:

- You have a basic understanding of the Motif™ user interface styles; that is, you can find your way around an application written with the Motif look and feel. (Check your local technical bookstore for Motif user interface books)
- You have installed the NVR Development Kit. (See Quick Install on page x or Installation Guide in Appending C.
- You have added `nvr/bin` to your path
- You have connected a video source device to one of the Parallax XVideo input ports



Display Controls



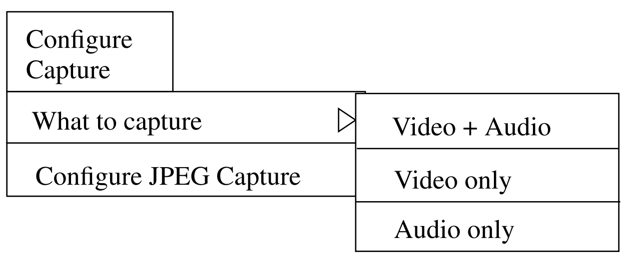
OK, you have a video image displaying in the video display. Once again, make sure that your video is active. Now pull down the **Display** item from the menu bar and select **Show Display Controls**. A dialog box will pop up which contains will allow you to make adjustments to the video image being displayed. The specific adjustments are dependent on your particular environment but may include **hue**, **saturation**, **brightness**, and **contrast**. You can use these controls to improve the appearance of your source material. Please note that *when you record, you capture the video image as it appears in the video display*, including these adjustments. Controls are also contained here for the audio monitor volume and for a choice of whether to output audio to the workstation's internal speaker or to a headphone jack.

of the same video resource as the recorder is using, the recorder is said to be *overridden*. You may wish to have the recorder take back control of the resource (**Restart on override**) or to allow the newer process to maintain control (**Stop on override**). The **Priority** dictates the priority of the video display activity with respect to that of other competing processes on the system. A priority of **Video High** will ensure the smoothest possible video display under all system load conditions, at the possible sacrifice in performance of other processes. A priority of **Video Low** will allow other system activities to perform better, but may cause discontinuities in the video display under certain system load conditions. *The availability of these control options is dependent on your hardware and software configuration*.

Now to Record

To record can be quite simple. First you select a destination file; the recording medium. To do this push the **Destination** button. A Motif *file selection dialog* will pop up for you to make your selection. Your file path specification will be displayed in the box underneath the destination button until you change your selection. Should your path be longer than the display box, you may scroll the text right and left by clicking on the text and dragging past the end of the box (this utilizes a Motif *text widget*).[1] If you have selected an existing JPEG movie file, you may append newly recorded material to the end.

To begin recording to your destination file, press the **Record** button. Recording continues until you press the **Stop** button. When complete, you will be given the results of your recording operation. That's it!

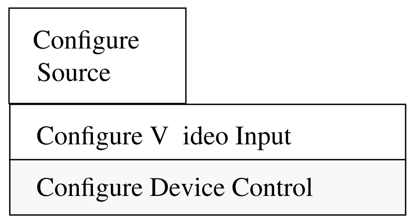


The recorder does offer additional control over the record process, should you wish to use it. You may choose to record only video or only audio. You may also adjust some characteristics of the JPEG compression process.

1. Motif documentation documents the details of each of their types of widgets. Those descriptions will not be repeated here.

Device Control

The recording method described above is simple, flexible, and supports the widest range of video source devices. Some video devices now support computer control of their behavior. Some, such as the Pioneer LD-V4400 laser disk player, allow direct control via an RS-232 connection. A host of others with LANC (formerly Ctrl-L) ports may be controlled using a Sony CI-1000 ViSCA Vbox between the workstation and the device. Before attempting the procedures described here, first make sure that you have made the proper physical connections to your equipment. Refer to the Device Connections section for more information.



To engage device control, pull down the **Configure Source** menu bar item and drag down to the **Configure Device Control** selection. This will open a dialog box allowing you to make the necessary selections. You will need to specify two pieces of information. One is the RS-232 port connected to the device (either ttya or ttyb). The other is whether you are connecting a Pioneer LD-V video disk player or a ViSCA/LANC-controllable device connected through a Vbox. Once you have made your selections, press the **OK** button to have them take effect. Some additional buttons will be added to the main window of the recorder application. Should you decide not to add or change device control, press the **Cancel** button to close the dialog with no effect.

point section. To set the in point, simply press the **In Point** button and the current counter value will be entered in the display in this row. The second row, the outpoint row, works in the same fashion. To verify that you have specified the segment that you would like to record, press the **Preview** button. If you have specified in and out points and the in point precedes the out point, the media will position to the in point and play the segment up to the out point and then pause the source device. You may stop the preview operation before completion by pressing the **Stop** button.[1] You may make fine adjustments to the in point by pressing the **<** or **>** buttons in the inpoint row. This will decrement and increment, respectively, the in point position in the display.[2] You may also set an entirely new in and/or out point by repositioning the media and again pushing the **In Point** or **Out Point** buttons.

To record the specified segment, first select the destination file, then press the **Perform**[3] button. Again the media will position to the in point and play the segment. This time it will be recording as it plays. As with the preview operation you may stop the operation before completion by pressing the **Stop** button.

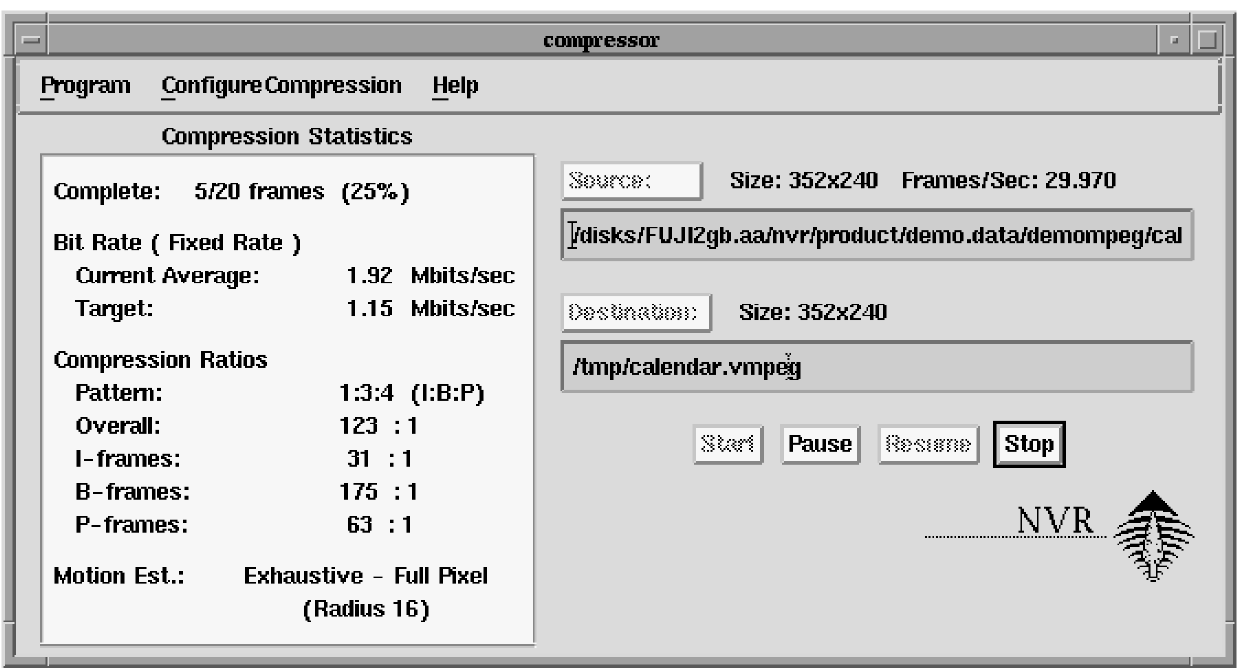
With ViSCA/LANC control, the media will actually position three seconds prior to the in point during the preview or segment record operation. This section, known as the *preroll* section, is played to allow the video tape image to stabilize before the in point is reached and the requested operation begins. The counter above the transport controls will reflect this.

1. On some device types, the step of positioning to the in point is atomic; the stop request will not take effect until the in point has been reached.
2. Not all devices are sufficiently accurate to position at a specific frame; all devices will position as closely as they are able to the position specified.
3. The label Perform for this operation derives from the same-labelled button on many video edit controllers for executing a pre-specified edit.

How To Compress Sequences Into MPEG

This section is a tutorial about how to turn movie sequences into MPEG video, audio and system streams using NVR’s `compressor`, `smpegcomp` and `ampegcomp` program. We make the following assumptions about what you know and have done before reading this:

- You have a basic understanding of the Motif™ user interface styles; that is, you can find your way around a Motif environment pretty well. (Check your local technical bookstore for Motif user interface books)
- You have installed the NVR Compressor or the NVR Development Kit which contains the compressor, smpegcomp, ampegcomp and other tools. (See Quick Install on page x or Installation Guide in Appending C.
- You have added `$NVRHOME/bin` to your path

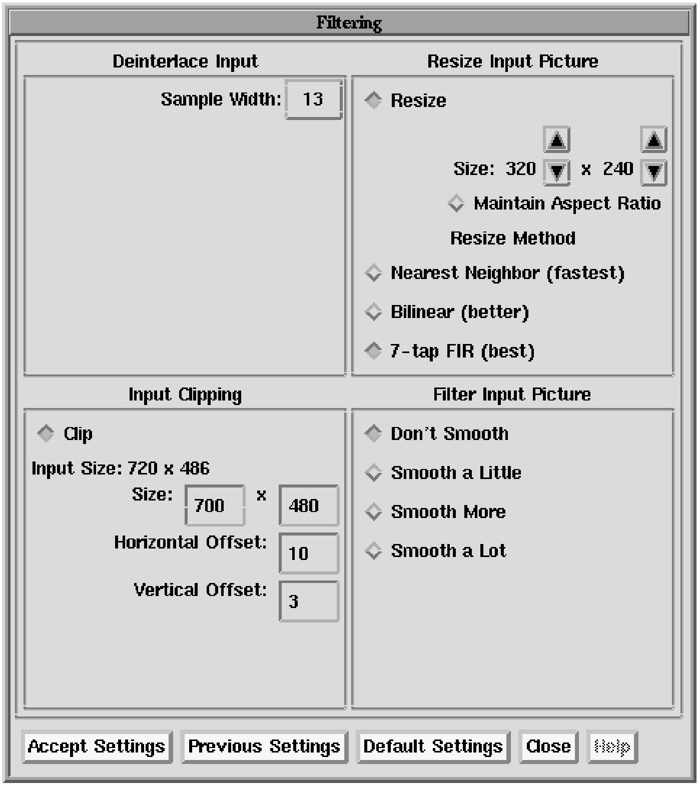


Pause button. No further frames will be compressed until the Resume button is pressed. Should you decide to abort the compression process before completion, press the Stop button. After a confirmation to make sure you did this intentionally, you will be given the option to:

- stop immediately and throw away all work done
- finish compressing the current group of pictures before stopping and leave the partially compressed file intact. This file will be a valid MPEG file viewable by the NVR player application.

Tailoring the Filtering

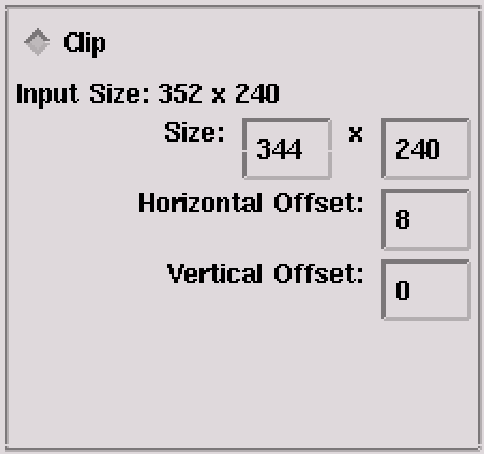
Before your movie data reaches the MPEG video compressor, you may want to change its size, clip off ugly borders, or perferm other pre-filtering in order to maintain the highest possible quality in the final MPEG sequence. To begin, select the menu item shown above and the filtering panel will appear as shown below.



Clipping

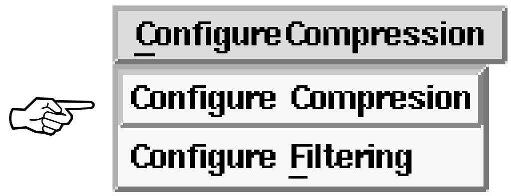
Frequently , a digitized video image will have a black border along one or more edges. This is caused by a mismatch between the video signal and what gets digitized. Whatever the reason, you can arrange to clip off any edge. The image below can improved by clipping about 8 pixels off the right hand edge.

The size of the picture above is 352x240, so this can be done by setting the horizontal offset to 8 and reducing the width by 8, down to 344 as shown below .

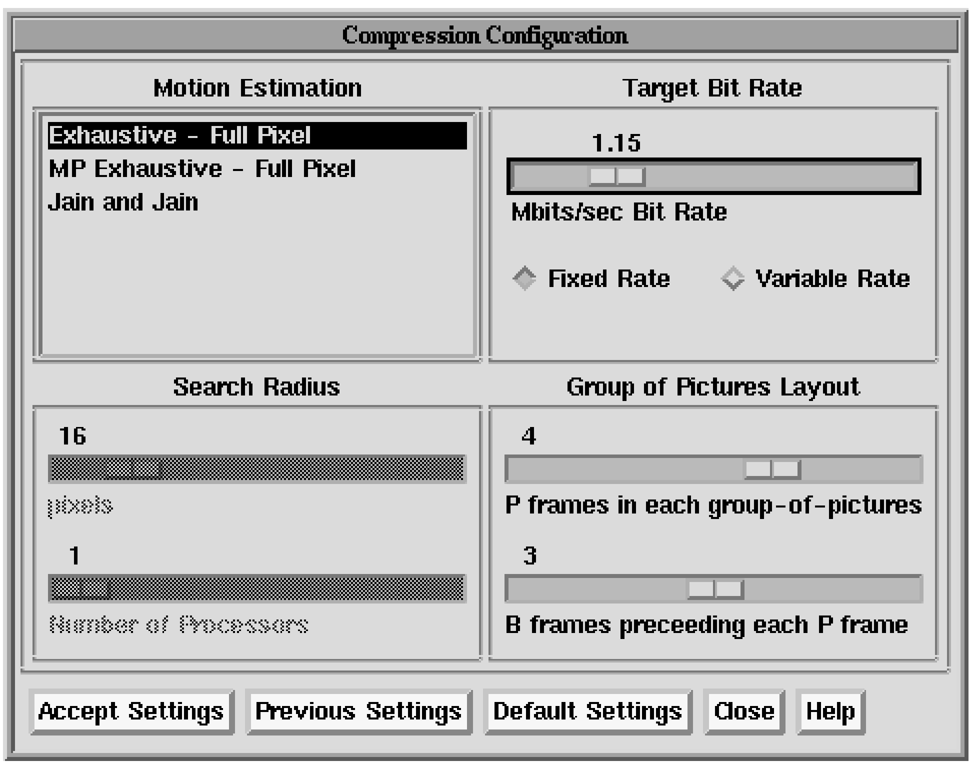


to having accepted them, press the **Previous Settings** button to return to the values that were in place when the configuration dialog was opened. To have the values set at the NVR recommended defaults, press the **Default Settings** button (this may require hitting the **Accept Settings** button to take effect). At any time, you may press the **Close** button to close the configuration dialog. If you have made changes, you will be asked whether they should be accepted before closing (equivalent to **Accept Settings**) or not (equivalent to **Previous Settings** and **Close**).

Tailoring the Compression



The MPEG standard offers some degrees of freedom as to the specifics of how a file is compressed. Adjusting some of the features of the compression process may allow you to better tailor the compressed MPEG file to the environment in which it will be used.



The MPEG algorithm gains much of its compression by utilizing a pared to the original, uncompressed frames. Maximizing the number of P and B-frames, however, will not guarantee better compression (and can even result in less compression under certain circumstances). You may also create a valid MPEG file with no P or B-frames. The compression gained in these types of frames is also the greatest source of reduced quality. I-frames also require less effort to decode.

An I:B:P ratio of 1:3:2 will result in the following layout:

I BBB P BBB P BBB ...

When you have finished making your compression selections, press the **Accept** button as described above.

Compressing Audio Into MPEG

The steps to compressing audio into MPEG are as follows:

Setting up the MPEG Audio Compression

- Select your audio material, making sure it is in raw, PCM form or AIFF, and that it is 32 kHz, 44.1 kHz or 48 kHz.
- Choose your desired bitrate and other attributes for MPEG audio.
- Compose a command line that will accomplish your goals

Note: At this time, the graphical user interface for the audio compressor is not available; only a command line version is available. Please see the `ampegcomp` man page at the end of this manual.

Converting Parallax Movie Audio Into Sun Audio

Parallax Movie Audio

If you are working with Parallax Movie files, `ampegcomp` can compress these into MPEG audio, or, you can convert the audio in these movie files to Standard sun audio files. Please see the `jmovieToAu` man page at the end of this manual.

Creating an MPEG System Stream

Creating a System Stream

When creating an MPEG system stream, the video and audio are said to be "multiplexed" together. The steps to creating a multiplexed MPEG system stream are as follows:

- Select a compressed MPEG audio sequence.
- Select a compressed MPEG video sequence.
- Choose the desired parameters for the system stream, such as packetsize.
- Compose a command line that will accomplish your goals

Note: At this time, the graphical user interface for the system mul-

How To Go Faster

This section is a discussion of possible approaches to getting better, faster smoother video for all stages of the video compression/decompression process.

Analog Video Playback

The design of the Parallax Xvideo board is such that most video activity does not interfere with graphics, and vice-versa. However, very intense graphic activity, like that done during software MPEG playback, may run slower if video is being digitized to the display at the same time.

The pause button

All that's necessary it to click the pause button in the `recorder` (and the `player` if you have another one to view analog video).

JPEG Recording

The recording process is accomplished by a separate server that is part of our Media Control Software (MCS) architecture, so it is only limited by the hardware and the lowest levels of system software.

Parallax has some well-designed hardware that we know is capable of recording 640x480 video at 30 frames per second with very high quality. However, their current release of software will not support this much activity. It is performance-bound by their Xlib library interface and by the X server's interface to the Parallax hardware. NVR is working to improve this performance bottleneck.

Fortunately, there are a few things that can be done to improve the performance. First, let us agree on a few parameters as a baseline for discussion. The recorder is normally used to capture frames that are 320x240, and the default quality factor[1] is 100.

Our studies have shown that frame capture through the current Parallax software is limited to about 24 frames per second, without disk output. Delays caused by disk output to a UNIX file system can reduce that to as little 15 frames per second.

Improvements to these numbers can be made in two ways:

- Reduce the size of each captured image

1. The quality factor is a scale applied to the recommended JPEG quantization tables. Refer to the section on Video Digitization for more detail.

guard against mistakes.

Third, use the SunOS disk administration utility, `format`, to assign new partitions. Your expert should be very familiar with this program. **If not, do *not* proceed**. In general, you should rearrange your partitions so that the raw partition is located at the end of the disk[1].

Fourth, restore your files. Do not make a file system on the raw partition or try to mount it.

Fifth, change the permissions on the block-special device[2] such that the user who runs the NVR `recorder` and `player` can read and write the disk. The person who moved the partitions will know the name of this device. We recommend that both the block- and character -special device be move to a different name so that no one comes along and makes a file system there.

Finally, start up the `recorder` and change the destination to be that raw partition. On our systems, we call it `/dev/movie`. The recorder will ask for confirmation before initializing your raw partition. After that, It will function just like any other movie file, only faster. The `player`, too, will recognize this special file after it contains recorded frames.

What improvement can a raw partition make?

The raw partition should get you much closer to the potential of 24 frame per second mentioned earlier.

1. SunOS stores the partition parameters and other configuration data in the first few cylinders. If you put your raw partition near the beginning of the disk, the system will work to begin with, but eventually will crash, and you will lose everything on that disk.
2. We only support the block-special device because it takes care of the operating system's buffer cache.

Hardware & Software Requirements

Sparc-based System Requirements

Overview

NVR software is available on a number of platforms. This section describes the requirements for each.

Sparc Hardware

NVR software has been tested on SPARCstation 2, SPARCstation IPX, SPARCstation IPC and Tatung CompStation 40.

Note that the IPC is very slow with respect to the other machines, and software MPEG playback may be too slow to be useful.

Display

Different NVR tools have different requirements for display hardware. These are the requirements for the release 2.0, rev F:

`Player`	For analog and JPEG playback, the Parallax Xvideo board is supported. For MPEG playback, most any framebuffer configured to look like a standard Sun 8-bit color frame buffer will work. We have tested CG3, CG4, CG8 and Parallax Xvideo.[1]
`Recorder`	The Parallax Xvideo board.
`Compressor`	Any display, color or monochrome, should work.

Operating System Software

NVR software tested under Solaris 1.0.1, also known as SunOS 4.1.2 and 4.1.3. The software may run under SunOS 4.1.1, but has not been tested in this environment. It does not run under Solaris 2.

In general, on the Sun, most of our software was built with static linking of libraries, so that it doesn't matter what dynamic libraries you have on your machine. All that matters is the underlying operating system.

Window System Software

We run OpenWindows 2.0 and 3.0 at NVR, but if you are using the

1. The Software MPEG playback system also works very well on 1-bit monochrome displays, but the user interface is not configured to allow it. If you need this feature, please contact us.

Device Connections

Cables and Boxes and Plugs...Oh, my!

This section concerns connecting video source devices, such as laser video disks, VCR's, and video cameras, to the Sun workstation for use by NVR application programs. For Silicon Graphics, please refer to their documentation . The connections covered here allow:

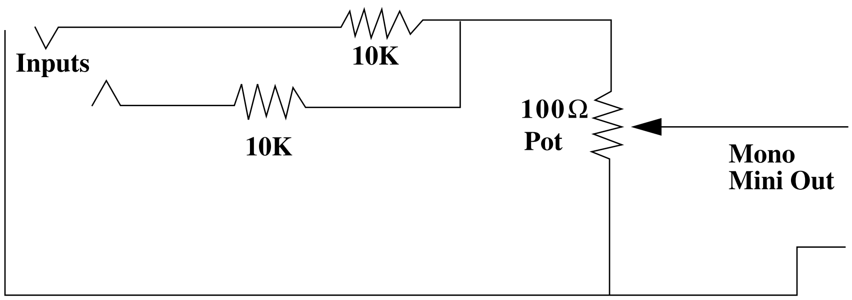
- audio to flow from a source device into the workstation
- video to flow from the source devices into the workstation
- the workstation to control the video devices

Audio Input Connections

The most common mistake in connecting audio from a source device to the workstation results from confusing the "level" of the audio signal. The workstation is expecting the audio input to be at "microphone (or mic) level". Most video devices, however, output their audio signal at "line level". And unlike many pieces of equipment these days, this is not necessarily apparent from the shape of the connectors.

Read the documentation for your specific video/audio source device carefully to determine the level of its audio. Connecting a line level signal to a workstation expecting a mic level input may physically damage the workstation's audio input electronics.

So... what do you do if your signal levels aren't compatible? You need to insert an attenuator inline to perform the signal level reduction. An attenuator is a simple device that reduces the signal from the higher line level to the lower mic level.



Schematic for a simple attenuator (-40 db ->)

software always utilizes the default protocol configurations (e.g. parity, data length, and baud rate) for the video devices.

Many other devices, usually in the class of video cameras and VCR's, are controllable via a connection known as LANC (formerly Ctrl-L). This type of control connection allows commands at the granularity of a remote control to be given to the video device via a physical connection. For this type of device you will need to purchase a Sony Vbox (CI-1000 Video/Computer Interface). This product connects to the RS-232 port of the workstation and to the LANC/Ctrl-L port of the video device[1]. The workstation communicates with the Sony Vbox using the ViSCA protocol[2]. In the NVR recorder and player applications, be sure to specify the device control of ViSCA/LANC in order to control a video device connected in this way.

Our Sun SPARCStation IPX utilizes 8-pin mini-DIN connectors for its RS-232 ports. We were able to plug the Sony Vbox RS-232 cable directly into either RS-232 port and communicate successfully. If your workstation uses another connector, such as the DB-25, you will need an adaptor cable to communicate with the Vbox.

The NVR software utilizing the ViSCA/LANC type of connection has been developed using the Canon A-1 video camera and further tested using a Sony TR-101 video camera. Many other video cameras and VCR's are also controllable via a LANC connection[3].

1. The NVR software requires that the Sony Vboxes be used individually and not be daisy-chained.
2. The Sony ViSCA protocol encompasses several low-level protocols. NVR software utilizes only the LANC/Ctrl-L capabilities.
3. We are very interested in hearing reports of successes or difficulties in using other LANC devices with our software. Our goal in choosing to support the LANC protocol is to support the widest possible array of video input devices.

Installation Guide

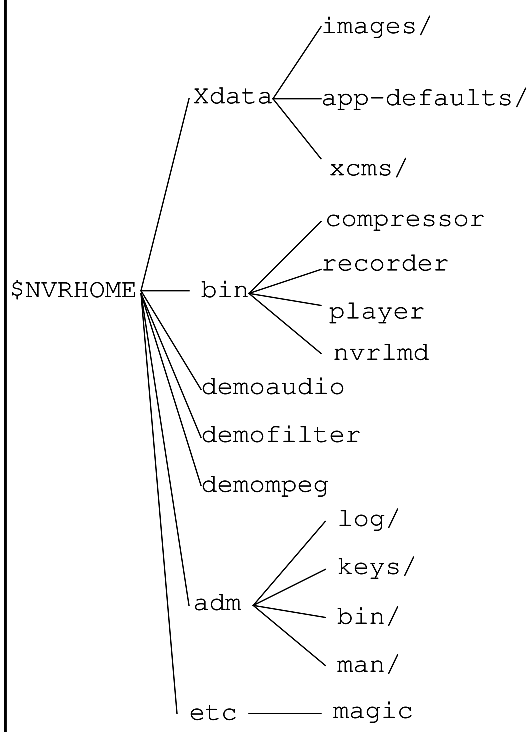
You should read this if...

This section is intended to give every aspect of the installation process in detail.

For most sites, the **Quick Install** section covers 90% of the instructions necessary to get up and running. A few sites will need a little more guidance. If you fit one of the descriptions below , you probably should read this guide before storing this manual with the rest of your software documentation:

- Your software doesn’ t run.
- You have several different kinds of monitors that will be displaying MPEG sequences
- You have more than one machine that will be using the software
- You don’ t have a fever but your temperature is far above normal

File Structure of the Software



NVR software can reside anywhere on your system. W e suggest a central place if more than one person will use the software, such as /

Environment Variables

There are three environment variables that you should modify in order to use the kit: `NVRHOME`, `PATH` (also known as `path` in the C-shell) and `XFILESEARCHPATH`. The changes suggested here should go into your `.cshrc` if you use C-shell and `.profile` if you use the Bourne shell.

Setting `$NVRHOME` environment variable

The most important environment variable is `$NVRHOME`. This tells the NVR software where to find other software pieces it needs. If this has not been set, the software may not run correctly. If you chose `/usr/local/bin/nvr` for the installation directory then `$NVRHOME` should be set as follows in the C-shell:

```
% setenv NVRHOME /usr/local/bin/nvr
```

And in the Bourne shell:

```
$ NVRHOME=/usr/local/bin/nvr
$ export NVRHOME
```

Setting your execution path

Your execution path should be set to include `$NVRHOME/bin`. This is done in the C-shell as follows:

```
% set path = ($NVRHOME/bin $path)
```

And in the Boure shell:

```
$ PATH="$NVRHOME/bin:$PATH"
$ export PATH
```

The variable `$XFILESEARCHPATH` can be used to tell X applications where to find X resources for a particular application, sometimes called application defaults. In our kit, we distribute the application defaults for three programs: the `player`, `recorder` and `compressor`. These are in:

```
Xdata/app-defaults/NVRPlayer
Xdata/app-defaults/NVRRecorder
Xdata/app-defaults/NVRCompressor
```

Setting `$XFILESEARCHPATH` environment variable

By changing your `$XFILESEARCHPATH` environment variable correctly , you will allow each of these programs to find their respective application defaults file. Under the C-shell, you type:

```
% setenv XFILESEARCHPATH \
$NVRHOME/Xdata/app-defaults/%N%S
```

Or for the Bourne shell:

```
NVRHOME=/whereever-you-put-it
if [ -f $NVRHOME/bin/nvrlmd ]; then
echo "`$NVRHOME/bin/nvrlmd -i` started." \
> /dev/console
$NVRHOME/bin/nvrlmd -m 0.25m -s0 \
-e $NVRHOME/adm/keys \
-l $NVRHOME/adm/log/nvrlmd.log \
> /dev/console
fi
```

To start up the license manager at boot time, you must add the lines to `/etc/rc.local`. It sets the log file to `$NVRHOME/adm/log/nvrlmd.log` and limits the log file size to 1/4 megabyte. If you need to start the license manager by hand, we have a simple shell script that does the same thing; simply type:

```
% startnvrlmd
```

When the license manager is running, NVR clients on the same subnet will be able to find the license manager by doing a UDP broadcast. For the uinitiated, this is like having someone paged over the intercom. The problem with both paging and UDP broadcasts is that they can only be heard a limited range. If your license server is not on the same subnet as the machine where you want to run a client, simply initialize the environment variable `NVR_LMHOST` to the internet name of the license manager host. For example, if your license manager host is `pear.fruit.com` but you want to run NVR applications on `orange.fruit.com`, then start the license manager on `pear.fruit.com` as shown above, and initialize your shell environment on `orange.fruit.com` with the command:

```
% setenv NVR_LMHOST orange.fruit.com
```

For more information, there are manual pages on the Élan software in Appendix D.

Known Problems and Solutions

This subsection describes the problems we have encountered with hardware and software. If we know of a solution, you can find it here too.

Parallax hardware JPEG compression is too slow

Parallax has excellent hardware capable of compressing 640x480 at 30fps at a very good quality. We have not yet integrated our software with the RTV toolkit, but we are working on it.

The JPEG frame displayed in the player is all gray.

Manual Pages

The following pages contain traditional UNIX™-style “man pages”. They should help you sort out the finer details of command line options for NVR programs.

the following types:

- AIFF
- AIFC (however , no compressed data)
- AU (Sun Audio)
- Parallax JMOVIE file

If the input file does not match the specified sampling rate, then it will be converted on-the-fly .

EXAMPLE

Given anAIFC audio file, named `file.aiff`, containing 16-bit, stereo, 44.1kHz audio, this can be compressed to an MPEG audio file named `file.ampeg` as shown below . Note that since not attributes were specified the attributes of the AIFF file (bits per sample, rate channels) will be the same as the input.

```
cvtaudio file.smpeg file.aiff
```

SEE ALSO

player , recorder , vmpegcomp, smpegcomp, playaudio, cvtaudio

COPYRIGHT

BUGS

The option flags will soon change to a `name=value` model.

The resampling method used to convert sampling rates is fast, but low quality .

Because the audio library used to read the input file can read any file type, it is easy (too easy) to specify MPEG audio as your input. It doesn't compress it any more, it just decompresses it and recompresses it. This will just make the quality get worse. Y ou should use the original material.

NAME

cvtaudio - audio conversion tool

SYNOPSIS

```
cvtaudio [attribute] ... <input> <output>
```

OPTIONS

rate= *N* Specifies an output rate of N; e.g., `rate=44100` or `rate=32000`.

bits= *B* Specifies the output bits per sample; e.g., `bits=16`. A special form is `bits=ulaw` or `bits=mulaw` which means 8-bit mu-law encoding (the number of channels must be 1, in this case).

channels= *C* Specifies the number of channels as C. Currently this should be 1 (mono) or 2 (stereo).

quality= *Q* Specifies the quality level for decoding the input; it must be a number between 0.0 and 1.0; e.g., `quality=.25`. This is only useful for MPEG input files.

type= *name* Specifies the output file type; e.g., type=aiff. Currently, the only output types available from this program are `au` (Sun Audio), `aiff` or `aifc`.

DESCRIPTION

This program converts takes an input file containing audio and produces an ouput audio file.

Input files may be any of the following types:

- AIFF
- AIFC (however, no compressed data)
- AU (Sun Audio)
- MPEG audio
- MPEG system (containing MPEG audio)
- Parallax JMOVIE file

EXAMPLE

Given an MPEG system stream, named `file.smpeg`, containing an MPEG audio stream, this can be converted to an AIFF file named `file.aiff` as shown below. Note that since not attributes were specified the attributes of the AIFF file (bits per sample, rate channels) will be the same as the input.

```
cvtaudio type=aiff file.smpeg file.aiff
```

Given the same MPEG system stream, if you want to convert the audio to 16kHz mono, the command line would be:

```
cvtaudio type=aiff rate=16000 channels=1 file.smpeg
file.aiff
```

SEE ALSO

ampegcomp, playaudio

COPYRIGHT

NAME

Imageop - Process sequences for use with an mpeg compressor

SYNOPSIS

```
imageop [op args...] [op args...] ...
```

OPERATORS

io

resample

filter

color

deinterlace

ARGUMENTS for OPERATORS

io:

read or read=<filename>
type=abekas|rawycrcb|rawycbcr|rawrgb|jmovie|sgirgb|cperfile
|avs|vicar|xwd|gif|rle|jpeg|miff|mtv|pnm|sun|tiff|avi|nvr_idx
size=WxH
sample=420|422|444
prefix=<file name prefix>
suffix=<file name suffix>
ndigits=<number if digits between prefix and suffix>
start=<first file name counter>
stop=<last file name counter>

resample:

type=rawycrcb|rawycbcr|rawrgb
size=WxH
inclip=WxH+X+Y
outclip=WxH+X+Y
sample=420|422|444
method=nearest|bilinear

filter:

inclip=WxH+X+Y
WxH=<filter matrix>
decimate
decimate=vertical|horizontal
interpolate=vertical|horizontal

deinterlace:

width=n
method=copy|copyfield2|black|spatial

color:

type=rawycrcb|rawycbcr|rawrgb

DESCRIPTION

	unix compress and uncompress. The .z and .gz suffixes imply gzip and gunzip.- See implicit and explicit read and write file list specifications below .
`sample`	Specifies the sample rate of the image. Sample rate for rawrgb is always 4:4:4. Y/Cr/Cb images may be 4:4:4 or 4:2:2 or 4:2:0.
`size`	Specifies width and height of image; e.g., `size=320x240`. Ignored for files that contain size information.
`type`	There are several image and sequence types recognized. An important aspect is that each type implies a color space: RGB or Y/CR/CB. That color space becomes an internal attribute of the operator and can only be changed by adding a `color` operator to change color space. For read io operations, the filetype can be discovered from the file header . However , some file types have no header and must be specified with this operator: `abekas, rawycrcb, rawycbcr, rawrgb, sgiyc, cperfile`. Unless inherited from previous operators write io operators must have an explicit file type. See FILE TYPES for a complete list.
`width`	Specifies the width of the spatial deinterlace filter .
`WxH=<filter>`	Used with the filter operator , this specifies an arbitrary FIR filter . Filters can be 1xN, Nx1 or NxM. Filter coefficients are specified as a list of numbers followed by a divisor . For example, the operator

```
filter 1x7=-12,0,140,256,140,0,-12/256 \
interpolate=vertical
```

will implement the MPEG 2 recommended interpolation filter

FILE TYPES

abekas	Image dumped from an ABEKAS D1 machine in CCIR601 format (720x486, 4:2:2, interlaced). This is also the same as Àccom CCIR601 format
avi	Microsoft A VI movie file format (RGB subset only).
avs	Advanced V isualization System image file format (not heavily tested)
cperfile	Component per file; i.e., Y , U, V like that used in Andy Hung' s compressor (from Stanford). U corresponds to CB and V corresponds to CR.
gif	Graphic Image Format (read-only)
jpeg	Joint Photographers Experts Group.
jmovie	Parallax JPEG movie file (read-only)
miff	Machine Independent file format (from ImageMagick package)
nvr_idx	NVR index file. ASCII file used to specify lists of image files. See the nvridx manual page for details of the format. (read-only)
pix	Alias' s RGB file format.
pnm	Portable bitmap, aka PBM or PBM plus.
rle	Utah Raster T oolkit; read only .

NAME

Mpegsmux - MPEG 1 system stream multiplexer

SYNOPSIS

mpegsmux <global options> <per-stream options>

In almost all cases you will only need to use a small number of options to make this command useful. For example, to create a generic multiplexed stream named `foo` of an audio and video sequence named `seq.ampg` and `seq.vmpg`, the command below will do that:

```
mpegsmux stream=foo audio=seq.ampg video=seq.vmpg
```

If instead, you need to produce this for CD-i or Video CD 2.0, then simply add the format specification, like these:

```
mpegsmux stream=foo format=cdi audio=seq.ampg video=seq.vmpg
mpegsmux stream=foo format=videocd audio=seq.ampg video=seq.vmpg
```

The most interesting options are those that allow you to concatenate sequences (`videos=` and `audios=`) or add still pictures to Video CD 2.0 and CD-i output (`stills=`). All the other options should be considered as knobs and dials to "tweak" your sequence into something slightly different.

GLOBAL OPTIONS

stream= *file* — Name of the system stream output file

verbose — If this keyword appears, `mpegsmux` will provide verbose output about the contents of each sector. It also includes output about how the decoder would decode the stream. This can be very useful if you have problems with "buffer underflow" or "buffer overflow". Note that this only works for CD output, not generic multiplexing.

intialscr= *time* — Set the initial SCR value to "time" seconds, default is 0.

muxrate= *bps* — Set the muxrate to "bps" bit per second. If not specified 1.01 times the sum of the elemetary stream rates inflated by the packetization overhead plus the minimum pack header and system header data rates will be used.

ratebound= *bps*

Set the bound on the mux rate given in the system header; defaults to muxrate.

pad=0/1 — A "1" implies a padding stream to achieve a data rate exactly equal to the mux rate. In this mode the number of bits in the output file is proportional to the presentation time. This mode should be used if the target medium is a synchronous communication channel or storage device. If "pad=0" (default), then pack header containing a time stamp are used to terminate "gap in time" found between packet of the bitstream. This mode is useful for asynchronous networks and recording devices.

packtime= *time* — Will force a pack header to be omitted at least every "time" seconds. A pack header will also be sent following any channel idle period (regardless of pad-

For example, if the last still picture in a Video CD segment play occured at time 00:01:30:00 (or 90 seconds), and you wanted it to be displayed until 100 seconds, then the parameter `endtime=100` would do that. This argument is ignored if any elementary stream has a presentation time stamp that is later in time.

PER-STREAM OPTIONS

audio= *file*
: Introduces an audio stream from file “file”. Up to 32 distinct audio channels can be multiplexed and they are denoted by stream IDs of 0xC0, 0xC1, ... in the order they appear.

audios= *file*
: Same as audio= *file*, except that *file* is a list of audio sequences with optional time codes and EOR markers. See SEQUENCE LISTS below.

video= *file*
: Introduces a video stream from file “file”. Up to 16 distinct video streams can be multiplexed and the are denoted by stream IDs 0xE0, 0XE1, ... in order.

videos= *file*
: Same as video= *file*, except that *file* is a list of video sequences with optional time codes and EOR markers. See SEQUENCE LISTS below.

data= *file*
: Introduces a data stream from file “file”. Up to 16 different data streams can be multiplexed and they are enumerated by IDs 0xF0, 0xF1 ...

packetsize= *bytes*
: Gives the packetsize for this stream. This number includes the 6 byte packet overhead plus the optional fields of PTS, DTS and STD buffer size. If one of the optional fields is included, the number of payload bytes is decreased to keep the packet size constant.

bufsize= *bytes*
: Sets the STD buffer size for this stream. The default is 4096 bytes for audio & data and the VBV size plus one packet size (for multiplex jitter) for video.

bitrate= *bps*
: Sets the elementary stream rate (bits per second). This is a required parameter for a data stream; illegal for video and audio streams (smpegcomp extracts the bitrate from the audio and video stream syntax).

ausize= *bytes*
: Set the access unit size (and time) for a data stream; illegal for video and audio streams (this paramter is determined by partially decoding elementary audio and video streams).

aualign=0/1
: This flag specifies that a new access unit will be placed in the beginning of a new packet.

initialocc= *fract*
: This set the initial buffer fullness for an elementary stream.

maxbufsize= *bytes*
: Specify the maximum buffer size for this stream (defaults to the actual buffer size).

starttime= *sec*
: Specifies initial presentation delay in seconds. The default is 0.0. The stream synchronization procedure is as follows. First, additional delays for each stream are calculated without regard to 'starttime' so that the initial presentation times are concurrent at the largest presentation time stamp (PTS). Next, each stream is further delayed from that point by the ammount of its

if you have more than one still picture stream.

STILL PICTURE FILE FORMAT

The format consist of three fields separated by white space on a single line. The # character introduces a comment; the remaining part of the line is ignored. The fields from left-to-right are timecode, low-res picture, high-res picture. The timecode is specified in hour:min:sec:-frame (the meaning of frame comes from the `tcfps` parameter). The second and third field specify file names that contain the still frames; the path may be relative or absolute (preceeded by a '/' character). The second field is required; the third field is also is optional. For example:

```
# still picture segment play list
00:00:00:00    pict1-low   pict1-high
00:00:05:14    pict2-low
00:00:10:00    pict3-low   pict3-high
00:00:15:05    pict4-low
```

Although Video CD and CD-i specify that pictures be encoded as variable bit-rate, certain picture rates, and so on, most of these are "fixed up" by `mpegsmux` during the multiplexing process. A few cannot, such as picture size and buffer size, because it affects the encoding of the entire sequence. `Mpegsmux` will tell you what it can and can't fix.

Note that although CD-i allows up to 16 still picture sequences, mpegsmux can only handle one at this time. You can have as many pictures in the still sequence, but there can only be one sequence.

ABOUT AUDIO SEQUENCES...

As with video, audio streams in a generic system stream have only one limitation: it must be encoded as a fixed bitrate audio stream. This is not a requirement of the MPEG standard, but it is a practical limitation for this software.

CD-i and V ideo CD 2.0 formats have a few limitations for audio sequences. The important ones are summarized in the table below . There are a few other limitations, but these are taken care of by specifying `format=cdi`, `vcdseg` or `videocd`. Consult the standards documents for details.

	CD-i (`format=cdi`)	Video CD Segment Play (`format=vcdseg`)	Video CD Audio/Video (`format=videocd`)
Audio coding layer	Layer I or Layer II	Layer I or Layer II	Layer II only
Bit rate	any	224 kBits/sec.	Mono: 64 kBits/sec. 96 kBits/sec. 192 kBits/sec. Stereo, etc.: 128 kBits/sec. 192 kBits/sec. 224 kBits/sec. 384 kBits/sec.
Sampling Fre-quency	44.1 kHz	Same	Same
Emphasis	None or 50/15 µsec.	Same	Same
Number of streams	32	1	1

ABOUT SEQUENCE LIST FILES...

A sequence list file is a way to put several sequences together as if they are one. This can be a little confusing because the MPEG System Stream specification provides a similar capability , but its not quite the same. A generic MPEG system stream may contain up to 32 audio streams and 16 video streams, each with a different stream id. The intent is for these streams to be played in parallel. In contrast, a sequence list file allows you to take several video or audio sequences and and encode them serially (one after the other) in a single video or audio stream. This is especially helpful if you encoded a movie in "chunks" which is sometimes the case with long movies because of the vast disk storage requirements.

A sequence list file simply allows you to list each chunk (or file), one file per line. In addition, if you want a pause between each sequence, you can specify a start time for each file. V ideo

Although the MPEG standard allows it, variable rate video and audio sequeces are not allowed in generic system streams. CD-i and Video CD standards do not allow variable rate streams.

eNAME

nvradmin - NVR license management administration program

SYNOPSIS

`nvradmin` *[options] [features|alias|feature=alias,...]*

DESCRIPTION

Nvradmin is NVR’ s license administrator program (based on the Elan License Manager from Elan Computer Group, Inc.) for use with NVR applications. It’ s functions include instal- ling license keys, listing the users of licensed products, and other license administration functions. Those options marked with an asterisk (*) may only be invoked by the owner of the current license server process, by the super -user , or members of group **wheel** .

OPTIONS

-c Creates a license key file. *Nvradmin* will print a *server code* which the customer gives to NVR who, in turn, will supply a key .

If an *alias* appears as the final argument, then that argument will be installed as the *alias* for that feature. A suggested *alias* accompanies every key distributed by NVR. Subsequent commands such as **`nvradmin -l`** will print the *alias* instead of a numerical feature name. By default, the key file is placed in the key directory /usr/lib/elm unless the -e option described below is used. In either case, the key directory is not created, but should already exist.

Here’ s an example command line for creating a key file for NVR’ s Digital Media T echnology Kit that will reside in the directory $NVRHOME/adm/keys:

```
nvradmin -c -n -e $NVRHOME/adm/keys NVRDevelKit
```

-e *keydir* This option is only meaningful when the create options **-C** or **-c**, described above, are used. By default, license keys are placed in the directory /usr/lib/elm. The -e option specifies a different key directory . Thus,

```
nvradmin -c -n -e $NVRHOME/adm/keys NVRDevelKit
```

will cause the license key file to be created in the directory $NVRHOME/adm/keys. Please note that this directory must already exist - nvradmin does not create it.

-i Prints the release number of nvradmin and the release of the ELM library it was compiled with.

-k * Kill license server daemon *nvrlmd* . The daemon will gracefully terminate upon receipt of the kill request.

-l Lists the current users and outstanding licenses available for all features known to all license managers on the network. Known features are those features used since the daemon was started (usually the last time the system was rebooted.) If features are given on the command line, the listing is restricted to the features named.

A **D** in column one indicates that this client is connected but disabled (redun-

NAME

Nvridx - ascii file for use in describing lists of images an dtheir attributes.

ATTRIBUTE NAMES

read:

read or read=<filename>

type:

type=<see imageop for valid type names>

size:

size=WxH

sample:

sample=420|422|444

colorspace:

colorspace=yc|rgb

interlace:

interlace=true|false

framerate:

framerate=f(float)

nframes:

nframes=n(int)

prefix:

prefix=<file prefix>

suffix:

suffix=<file suffix>

ndigits:

ndigits=n(int)

start:

start=n(int)

stop:

stop=n(int)

DESCRIPTION

This is a movie index file. It provides info for the compressor UI for movies that have no 'headers", like raw data, and for collections of files that make up a movie.

This file type is identified by the seven characters `nvr_idx` at the front of the file. Following this are a number of `name=value` pairs describing attributes and their values. The comment character *#* is recognized anywhere in the file and causes the parser to ignore everything until after the next newline. There can be no whitespace within the name-value pairs but whitespace can be used anywhere in the file. Most of the attributes are required for the NVR `compressor` user interface to operate correctly on these files. *Framerate* , and *nframes* are optional but provide useful information to the UI. The nvridx file type is used primarily to describe collections of files to be compressed (as in a series of 10 *Tiff* files) or to describe raw

```
size=720x243       # ... the size
sample=4:2:2       # ... the sample rate, 422 works here also
interlace=true     # this file is interlaced
framerate=29.97    # frame rate of the material
nframes=60         # number of frames in the movie
```

grow a bit beyond the limit before it is backed up and truncated.

-s0 Start accepting license requests with zero delay. Without this option, the default delay is 3 minutes. This option should be used when starting the server for the first time. If you are restarting after nvrlmd crashed and clients are running, the 3 minute delay allows these clients time to renew their licenses first, before new clients start up.

-v # Select the verbosity of the messages printed to the log file. When a level is selected all levels less than or equal to the level are printed. Levels 5 and higher are considered debugging levels. Nvrlmd starts in level 3. The levels are:

1. Only error messages.
2. License failures.
3. License activity.
4. Client Connects and disconnects.
5. Message for every packet received.
6. Message for every packet sent.
7. More Client and Zombie information.
8. Info about key and resource file.
9. Major stuff.

EXAMPLE

Here is the suggested invocation of *nvrlmd* which would appear in the system startup file **/etc/rc.local**. It sets the log file to $NVRHOME/adm/log/nvrlmd.log and limits the log file size to 1/4 megabyte.

```
if [ -f $NVRHOME/bin/nvrlmd ]; then
echo "`$NVRHOME/bin/nvrlmd -i` started." > /dev/
console
$NVRHOME/bin/nvrlmd -m 0.25m -s0 \
-e $NVRHOME/adm/keys \
-l $NVRHOME/adm/log/nvrlmd.log \
> /dev/console
fi
```

If you are only using one machine to run NVR software, and you don't want to put this in `/etc/rc.local`, then the *player*, *compressor* and *recorder* will be able to start up *nvrlmd* for you. Just make sure that `NVRHOME` is set up in your environment.

ADMINISTRATION

When using a log file (highly recommended) you will need to trim it every so often. This can be done automatically with the -m option of nvrlmd as previously described.

SEE ALSO

nvradmin(1).

NAME

Player -User interface for viewing JPEG, MPEG or analog video

SYNOPSIS

`player` *`[options]`*

OPTIONS

-24bit	display in 24 bit color (MPEG only)
-12bit	display in 12 bit color (MPEG only)
-232	display in 7-bit color (MPEG only)
-332	display in 8-bit color (MPEG only)
-gray7	display in 7-bit grayscale (MPEG only)
-gray	display in same as gray7 (MPEG only)
-gray8	display in 8-bit grayscale (MPEG only)
-555	display in 125-colors; faster than -232, but more memory (MPEG only)
-666	display in 216-colors; faster than -332, but more memory (MPEG only)
-analog	Specifies analog video as the initial viewing mode. The serial port and protocol can be specified upon invocation via the -ttya, -ttyb, -lanc, and -ldv option; or dynamically via the Configure Source menubar item.
-jpeg	Specifies JPEG as the initial viewing mode.
-lanc	[for analog video] Specifies the use of Lan-C protocols for controlling the video device. This should be used in conjunction with the -ttya or -ttyb option.
-ldv	[for analog video] Specifies the use of Pioneer LDV protocols for controlling the video device (i.e., the Pioneer LDV family of video disc players). This should be used in conjunction with the -ttya or -ttyb option.
-mpeg	Specifies MPEG as the initial viewing mode. V iewing modes can be changed dynamically view the Select Source menubar item.
-path	[for JPEG and MPEG] Specifies a default directory in which to locate JPEG or MPEG files to view .
-srcFile	[for JPEG and MPEG] Specifies an initial file to load, ready for playing. If the srcFile is relative (no leading '/', then it is concatenated with the argument to -path, if any .
-ttya	[for analog video] Specifies /dev/ttya as the serial port used to control a video device.
-ttyb	[for analog video] Specifies /dev/ttyb as the serial port used to control a video device.

DESCRIPTION

This application enables you to play analog video, JPEG, and MPEG sequences on a Unix workstation. There is actually three different interfaces, one for viewing each type of digital

NAME

playaudio - audio playback tool

SYNOPSIS

```
playaudio [quality=Q] <input> ...
```

OPTIONS

quality= *Q* Specifies the quality level for decoding the input; it must be a number between 0.0 and 1.0; e.g., `quality=.25`. This is only useful for MPEG input files.

DESCRIPTION

This program plays back a list of input files containing audio. If the attributes of the audio do not match the hardware, on-the-fly conversion is done to get it into the correct type. Input files may be any of the following types:

- AIFF
- AIFC (however , no compressed data)
- AU (Sun Audio)
- MPEG audio
- MPEG system (containing MPEG audio)
- Parallax JMOVIE file

Input file types may be mixed on the command line. All input files are played back-to-back.

EXAMPLE

Given an MPEG system stream, named `file.smpeg`, containing an MPEG audio stream, this can be played as shown below .

```
playaudio file.smpeg
```

SEE ALSO

ampegcomp, cvtaudio, player

COPYRIGHT

record.

NVRRecorder*engine : <e>

Specifies the default engine; the conenctor of the frame buffer to which the video signal to be recorded is connected. The engine <e> is in the range <0..1>.

NVRRecorder*imagePath : <path>

Specifies a directory containing .miff files used by the recorder application.

NVRRecorder*sourceDevice : <sd>

Specifies the default type of source device control.
<sd> = 0 specifies nocontrols/camera mode
<sd> = 1 specifies Pioneer LD-V
<sd> = 2 specifies V iSCA/LANC

NVRRecorder*sourceDeviceTty : <tty>

Specifies the default tty used for control of video source device.

SEE ALSO

player , compressor

-P*n* For the -M3 option, this specifies the number of processors to use during motion estimation.

-q*n* Define a fixed scale *n* (1 to 31). Fixed bit rate is disabled.

-o<file descriptor>
Send compressor events to this file descriptor (stderr default). After every frame compressed, the compressor sends separate events specifying frame number, bits per picture, and compression ratio. Generally, this is never used by the end user.

-s*s* Specify a slice size s, measured in macroblock widths. The default is a value equal to the length of one row of macroblocks. For example, a picture 320 pixels wide is 20 macroblocks wide.

-Bi,p,b Specify the initial complexities of I, P and B pictures as described in the MPEG 2 document. Default values are 160.0/1 15.0 for I, 60.0/1 15.0 for P and 42.0/1 15.0 for B.

DESCRIPTION

This program is a server/engine used by `compressor`. It is documented here for the customers that have requirements not supported by the graphical user interface.

`Vmpegcomp` is an MPEG encoder/compressor that takes a variety of video data and produces an MPEG video data stream. The `imageop` man page describes all of the image pipeline operators and their arguments. The same usage applies to `vmpegcomp`, except that the final pipeline operator and arguments must be:

```
... io write=filename type=mpeg
```

This is followed by the compressor options (I know this is ugly, soon it will all be symmetric). The only constraint on the input to this final pipeline member is that it be4:2:0 YCrCb. The pipeline operators make this simple. For example, to compress 100 sgirgb files named `foo.001.rgb`, `foo.002.rgb`, `foo.003.rgb`, etc., to a fixed 1.15 megabit stream:

```
vmpegcomp io read prefix=foo. suffix=.rgb \
      ndigits=3 start=1 stop=100 \
   color type=rawycrcb \
   resample method=bilinear sample=420 \
   io write=foo.mpg type=mpeg -M0 -b1150000
```

SEE ALSO

imageop, smpegcomp, audiocomp, compressor

r

s

t

v